uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática

Curso académico 2017-2018

Trabajo Fin de Grado

PREDICCIÓN DE LA RADIACIÓN GLOBAL UTILIZANDO REDES DE NEURONAS ARTIFICIALES

Álvaro Robledo Quintero

Tutores:

Inés Mª Galván León
Ricardo Aler Mur

Resumen

La predicción de la radiación solar es un campo de investigación que tiene un gran interés hoy en día y dónde se han probado gran cantidad de técnicas para predecir los distintos tipos de radiación de una manera precisa. Algunos de estos modelos presentan mejores resultados para horizontes de predicción a corto plazo, mientras que otros han mostrado buenos comportamientos a medio y largo plazo. En el presente trabajo se estudia la aplicación de técnicas de aprendizaje automático, concretamente redes neuronales, para la predicción de la radiación solar global horizontal y directa en horizontes tomados en intervalos de 15 minutos con un límite de hasta 6 horas en el futuro. Para el estudio se han utilizado datos de radiación global y directa relativos a una planta de Sevilla. Los resultados obtenidos con el Perceptron Multicapa serán comparados con los obtenidos por modelos utilizados habitualmente en el campo de la radiación solar, SmartPersistence y Satélite, así como con modelos de regresión lineal.

Observando los resultados obtenidos por las los distintos modelos, se puede concluir que para problemas de predicción de radiación global, las redes de neuronas, como norma general, obtienen mejores resultados que el resto de los modelos. En el caso de la predicción de radiación directa, el comportamiento de los modelos neuronales obtendrá mejores resultados que Satélite, pero por lo general no mejora las predicciones hechas por SmartPersistence. En la mayoría de los casos la generalización de las redes neuronales será mejor que los modelos lineales.

Palabras clave: Solar irradiation, Machine Learning, Neural networks, Multilayer Percetron, Global Irradiance Forecasting.

Abstract

Solar irradiaton forecasting field is getting great interest nowadays and large number of techniques have been tried to predict different types of irradiation in accurate way. Some of these models present better results for short-term forecasting horizons, while others have shown good behavior in the medium and long term ones. Along the present work we will study the application of machine learning techniques, specifically neural networks, to forecast horizontal and direct global solar irradiation in horizons taken every 15 minute up to 6 hours in the future. Along this Thesis, global and direct radiation data related to a Seville plant have been used. The results obtained with Multilayer Perceptron will be compared with those obtained using models commonly used in solar irradiation field, SmartPersistence and Satellite, as well as with linear regression models.

Results obtained by the different models show that, for global irradiation forecasting problems, neural networks models, as a general rule, reach better results than the rest of the models. In case of direct irradiation forecasting problems, the behavior of the neural models will obtain more accurate results than Satellite, but in general they do not improve the predictions made by SmartPersistence. In most of the cases generalization made by neural networks will be better than linear regression.

Key words: Solar irradiation, Machine Learning, Neural networks, Multilayer Perceptron, Global Irradiance Forecasting.

Dedicatoria

"The wolf on the hill is never as hungry as the wolf climbing the hill".

Arnold Schwarzenegger

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introduction

In this chapter an introduction to the problem of solar irradiance forecasting will be presented as well as the structure and objectives of the work.

1.1. Solar irradiance (GHI and DNI)

Solar irradiance is one of the main areas about energy networks nowadays. It is one of the cleanest renewable energies available today, therefore, due to the exponential growth of photovoltaic plants, a reliable forecast of the radiation values is fundamental for a better positioning of the plates regarding to the incindence angle of the solar rays, thus, energy produced by the photovoltaic cell could be maximize.

So, to understand solar irradiarion, it is necessary to present its main components:

- **Direct solar radiation**. Irradiation type which comes to the surface of the earth without suffering any change on its trajectory.
- **Diffuse radiation**. Irradiation type which has suffered some kind of change on its way to the earth's suface.
- **Albedo radiation**. Solar component reflected by some kind of surface (ground, snow, etc).

Types of solar irradiance that will be used in this work are:

- **GHI (Global Horizontal Irradiance):** Summatory of both direct and diffuse radiation.
- **DNI (Direct Normal Irradiance):** Type of diffuse radiation with a normal angle to the Surface. The moment of the largest amount of energy production by a photovoltaic plate is when solar rays strike perpendicularly.

Solar irradiance compontents are shown on the next picture 1.1

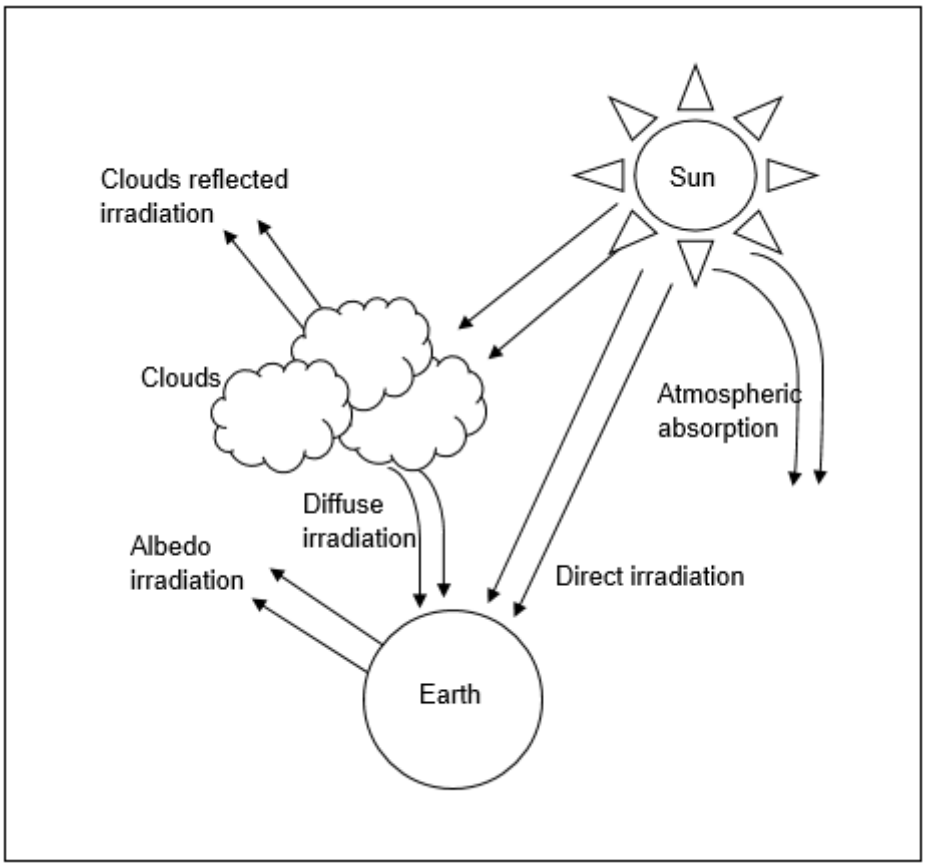

Fig. 1.1. Irradiation components

Within the solar radiation forecasting context, it is usual to tell apart between three main types of forecasting: Nowcasting, Short-term forecasting and Long-term forecasting. Nowcasting refers to a very near future prediction (up to 10 minutes). Shot-term forecasting refers to predictions made in horizons within 15 minutes up to 6 hours. Finally, long-term forecasting is relative to predictions made within 1 day up to a week. In this work, Short-term forecasting will be used.

1.2. Studies related to solar energy forecasting

As stated above, due to the exponential growth in the use of the renewable energies, the consolidation of photovoltaic plants into the electricity network plays a major part in our country. Because of this growth, the necessity of discovering more accurated forecastnig models towards developing new strategies of energy production, scheduling and distribution increase.

According to the reference [1], different groups of models can be distinguish within the GHI forecast problems:

The first model applies to the nowcasting prediction. There are two differents ways to make this kind of models:

- In one hand, applying AI (or statistical) models to real time meassurements of solar radiation, future values could be predicted with high accuracy.
- On the other hand, there are some classification models able to predict the irradiance in a distributed way from clouds pictures taken by a camera.

The second model is known as Short-Term forecasting. This kind of models are related to predictions within 15 minutes and 6 hours, and they are based on images taken by satellites. From the treatment of those images, the path followed by the clouds can be predicted so their future possition could be known.

To forecasting models in horizions within 3 and 72 hours, numeric methods of weather prediction (MNPM) ([2]) are used. With some of this type of models the horizon can rise up to ten days in the future.

Finally, it is possible to forecast radiation up to one month in the future using MNPM based models.

Facing all those types of solar irradiance forecasting methods, depending on the prediction horizon, some methods result in better accuracy than others, since certain models present better behaviors in short-medium term horizons than in the long term

Machine learning techniques have played an important role in the construction of models for the solar irradiation forecasting (see teferences [3] and [4]). In [5] neural networks are used for solar irradiance time series forecasting.

There are different approaches of the solar irradiation forecasting problem. Taking ([5]) as reference, neural networks are used to forecast irradiation taking the time series historical as input. In case of ([6], [7]), Support Vector Machines, SVM (see [8]) models are used. Support Vector Machines models are sort of supervised learning techniques which make models by building hyperplanes onto the input space. SVM are prepared to provide linear (linear kernel) and non-linear (non-linear kernel) approaches.

Machine learning techniques have also been used in forecasting problems to precidt solar irradiation taking as input weather variables coming from Numerical Weather Prediction (see [9] and [10]).

This Bachelor Thesis has been developed upon the research proyect "Aprendizaje y optimización evolutiva para predicción integrada de radiación solar", where some iradiation works has been carried out. Within those works, [11], use different regresion methods to estimate solar energy, Support Vector Machines, SVM, and gradient boosted regresión, GBR, [12]. Specifically, those methods are characterised by building decisión trees based models. From this previos study it is concluded that the non-linear techniques used obtain lower error meassures than the linear ones.

Other study that uses machine learning models to forecast solar irradiation (also developed upon the same research proyect mentioned before) and which has been taken as reference along this Thesis is [13]. This Project propose merging physical models using Extrem Boosted Gradient (reference [8]), a variation of Gradient Boosting. The approach used is summarized in the construction of a general model that takes as input the predictions made by the physical models mergin the data from every horizon in the problem, in order to minimize the global error. In addition, another model is constructed by training different models for each horizon. The last one, is more related to the present work, since they deal with time horizons individually. The results show a considerable improvement in the prediction accuracy over physical models in nearby horizons, and a similar behavior for more distant horizons.

With all this, the aim of this project is to study how accurate are the predictions using certain non-linear machine learning models. Specifically the Multilayer Preceptron, over the solar radiation domain, compared to the accuracy of linear models (simple regression)

1.3. Social and economic environment

Photoviltaic energy is one of the main renewables energies in Spain nowadays. According to the reports provided by Red Eléctrica Española, the total amount of demanded energy in Spain in 2017 was 268.5 TWh, where 8,350GWh of them were produced by photovoltaic methods. Comparing it to data provided 5 years ago, in 2012 solar energy produced a demand of 7,803GWh, which represents 0.7 % increase.

Even though this amount of energy may seem small, it is necessary to study better methodologies to foreacst solar irradiance, thus energy produced by photovoltaics cells could be maximized resulting on a substantial price drop of this technologies and a way to promote responsible policies with the environment.

1.4. Regulatory Framework

To develop the current Bachelor Thesis free software tools has been used (see 4.1), and MATRAS group from the University of Jaén (http://matras.ujaen.es/) has provided the necessary data.

This thesis has been developed within "ENE2014-56126-C2-2-R Aprendizaje y optimización evolutiva para predicción integrada de radiación solar" project which is part of a research project coordinated by the group stated above and it will run until the end of 2018.

1.5. Thesis goals

The aim of this Thesis is to study the behavior of machine learning models (also called statistical models in the field of radiation prediction) to forecast GHI and DNI within different prediction horizons. Along this work, the study has been carried out using neural network models, specifically, Multilayer Perceptron, and linear models such as linear regression.

Those models were made using a time series historical (GHI and DNI meassures). So, in first place, it is necessary to define how many previous instants will be taken as input to the statisticals model. Therefore, the **first goal** of this Thesis is to make a study based on the correlation between the historical data and the value to be predicted. This study will be carried out through the autocorrelation function of the target variables.

Once the number of previous instants to be used is defined, the **second goal** is to build a model using the Multilayer Perceptron (MLP). Given that, the network's behavior is highly dependent on certain hyperparameters (hidden neurons, learning ratio and number of cycles of training), in order to achieve this objective, an exhaustive search of values for these parameters has been carried out, deciding the best set of hyper-parameters using a group of different examples from those used in the stages of training and test, called validation set.

Finally, with the aim of meassure the performance of the neural networks, the **third goal** is to build a statistical model of linear regression using the same historical that has been used by the net. As it is well known, the MLP can build non-linear relations, so with this in mind we'll try to study if a non-linear approach provides better prediction results in the current problem than a linear approaches.

Behavior obtained by the statistical models will be compared to Smart- Persistence (model usually used in irradiation forecasting problems) and physical models based on indexing satellite images, advection and diffusion of cloud classifications, and uses mathematical models based on the physical principles of the atmosphere and oceans to make future weather predictions from the current climate conditions. These are: Satellite, CIADCast and WRFSolar respectively (see [13]).

To develop this Thesis, radiation data measured in a plant in Seville (see 3.1) are available and provided by the MATRAS group of the University of Jaén (http://matras.ujaen.es/). The study will be made in time horizons from fifteen to fifteen minutes with 6 hours limit in the future (short-term prediction). Therefore, the models studied in this project will be used to predict a total of:

$$\frac{6 * 60}{15} = 24 \text{ horizons}$$

1.6. Thesis Structure

The current work will be structured as follow:

- **Abstract**
- **Chapter 1: Introduction**. Along this section the problem of the prediction of two types of solar radiation: Horizontal Global Radiation (GHI) and Normal Direct Radiation (DNI) will be introduced. In addition, the main goals of the work carried out, the organization of the document itself and the planning followed during the development of the project will be presented too.
- **Chapter 2: Neural networks. Multilayer Perceptron.** This section of the document aims to introduce the theory that supports neural networks models from a global perspective. After that, the thoery of the models used along this thesis, Multilayer Percelptron, will be introduced
- **Chapter 3: Problem description and used dataset.** Detailed explanation of how the neural networks (Multilayer Perceptron) have been used to approach the solar irradiation problem using different time horizons, as well as a detailed description of the used dataset
- **Chapter 4: Experimental validation**. This chapter will make up the main body of the Thesis. Its aim is to describe the methodology used for the development of the prediction, as well as the software used and the results obtained. It is made of the following sections:
    - 4.1 Used methodology

- 4.2 Software
  - 4.2.1 RStudio and required libraries
  - 4.2.2 Used functions
- 4.3 Experimental results

- **Chapter 5: Conclusion and future works**. Closing chapter where the main findings acquired during the project development and the possible improvements or future works to be carried out from it are exposed

1.7. Planning: Gantt

In this section the planning followed throughout the Thesis will be described. For this purpose, three Gantt diagrams will be used, one every two months dedicated, where the tasks that have been carried out during the project and their duration in weeks will be indicated.

- Gantt 1: Schedule followed between January and February (see 1.2)
- Gantt 2: Schedule followed between March and April (see 1.3)
- Gantt 3: Schedule followed between May and June (see 1.4)

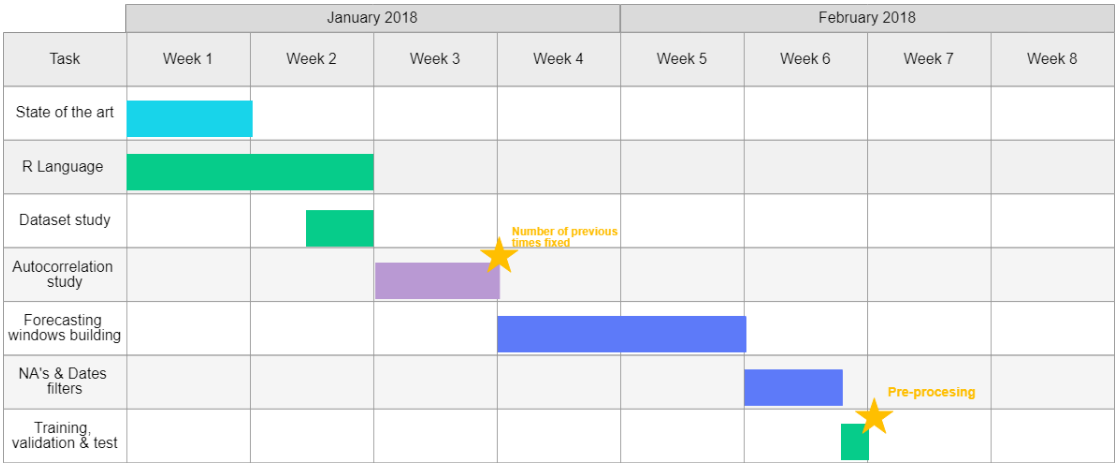

Fig. 1.2. Schedule followed between January and February

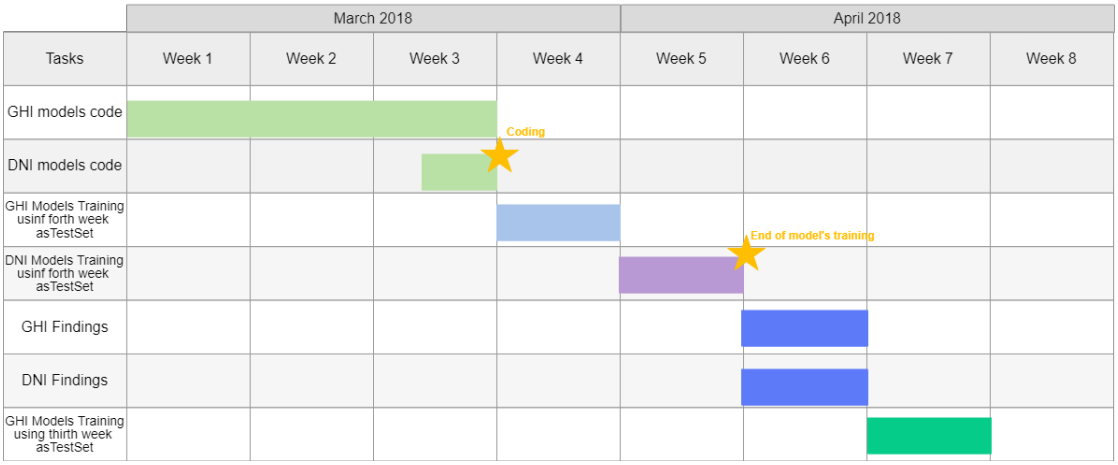

Fig. 1.3. Schedule followed between March and April

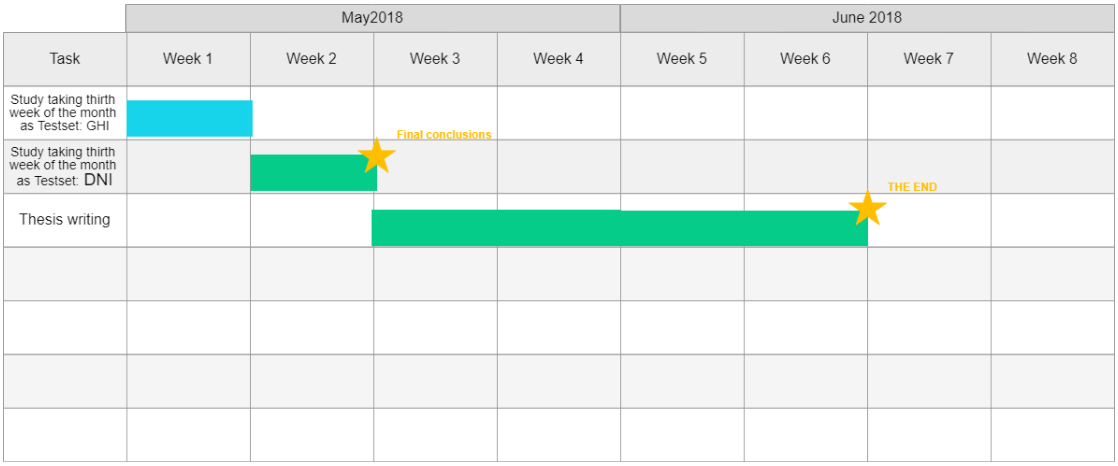

Fig. 1.4. Schedule followed between May and June

1.8. Budget

In this section a table (see Table 1.1) containing the breakdown of the project budget will be presented

Costs	Price	Time	Total	Description
Internet	65 €/month	3.5 months	227,00 €	ADSL 100 MB
Hardware Spending				
PC	1100 €/3 years	4 months	122,22 €	Lenovo G50, Intel Core i7, 8 GB RAM
Software Spending				
Tools Licences	0 €/year	4 months	0,00 €	RStudio
Microsoft Licence	77.84€/year	4 months	25.95€	Microsoft office 2013
Personal Spending				
Developer	23€/hour	704 hours	16.192,00€	Developer gross salary 88 days
TOTAL			16.567,67€	
Risk (30 %)			4.970,301€	
Gains (15 %)			2.485,1505€	
TOTAL WITHOUT TAXES			24.023,1215€	
TAXES (21 %)			5.044,855515€	
TOTAL WITH TAXES			29.067,98€	

Tabla 1.1. Budget

Capítulo 2

Redes de Neuronas. Perceptron Multicapa

Hoy en día la ciencia y tecnología se enfrenta a una gran cantidad de retos. Sin duda, uno de los más importantes es el de construir sistemas inteligentes. Se entiende por sistema inteligente a cualquier dispositivo, físico o lógico, que observa su entorno y reacciona con el fin de conseguir realizar una tarea específica.

En la rama de la ciencia denominada Inteligencia Artificial se pueden distinguir dos áreas principales. La primera es la encargada de generar sistemas con las características necesarias para poder definirlos como inteligentes (IA simbólica). En cuanto a la segunda, se encarga de construir sistemas que pueden ir evolucionando o adaptándose por sí mismos hasta poder ser capaces de encontrar una solución para el problema especificado (IA subsimbólica).

El punto de vista simbólico se caracteriza por el estudio de las herramientas utilizadas por los humanos para razonar ante un problema, cómo se enfrenta y cómo los resuelve. Una vez estudiado, se generan programas con la capacidad para emular estos patrones de conducta. Cuánto mejor se haya podido plasmar el modo de razonamiento, mejores serán los resultados del sistema inteligente a la hora de resolver el problema planteado.

Por otro lado, la perspectiva simbólica se caracteriza por el estudio de los mecanismos que definen a los seres vivos como inteligentes. El principal mecanismo que nos permite encontrar la solución a problemas complejos, y no preprogramados, es el sistema nervioso. Por lo que esta perspectiva se centrará en el estudio de dicho sistema, su funcionamiento, forma y características con en objetivo final de generar programas capaces de resolver una tarea.

El objetivo ideal de las Redes de Neuronas Artificiales (RNA) es poder diseñar sistemas que posean elementos neuronales de procesamiento en paralelo. De esta manera, el comportamiento de dicho sistema podría emular fielmente los sistemas neuronales de los animales. Esto hace que sea de vital importancia un estudio detallado de los mecanismos que guían el comportamiento de los sistemas de neuronas.

En los siguientes apartados del capítulo se describirán los fundamentos biológicos de las redes neuronales, así como diferentes modelos computacionales. Concretamente, se desarrollará el modelo utilizado en la realización del proyecto: Perceptron Multicpa.

2.1. Fundamentos biológicos de las redes neuronales

La red neuronal que poseen los seres vivos tiene como objetivo la recogida de información, su elaboración y transmisión. Este sistema se compone de tres partes:

1. **Receptores.** Están ubicados en las células sensoriales, recolectan la información en forma de estímulos, ya sea proveniente del entorno o de dentro del organismo.
2. **Sistema nervioso.** El cual recibe la información, la procesa, elabora y la envía a los órganos efectores.
3. **Órganos diana o efectores** (por ejemplo, músculos y glándulas), que reciben la información y la transforman en acciones

El elemento más importante del sistema nervioso, tanto de manera estructural como funcionales, es la neurona. Gran parte de éstas usan las sustancias que producen como emisión de información. Dicha información se manda a través de distintas estructuras con forma de red, dentro de las que se genera y guarda la información. Además, las neuronas tienen un componente que se relaciona con los receptores, los cuales captan señales del exterior.

Otra parte de la neurona dirige la información hacia los órganos efectores. Una de las prolongaciones se encarga de conducir los impulsos, a esta se la conoce como axón. Las neuronas se conectan unas con otras a través de un proceso llamado sinapsis (ver figura 2.1, tomada del libro [14]).

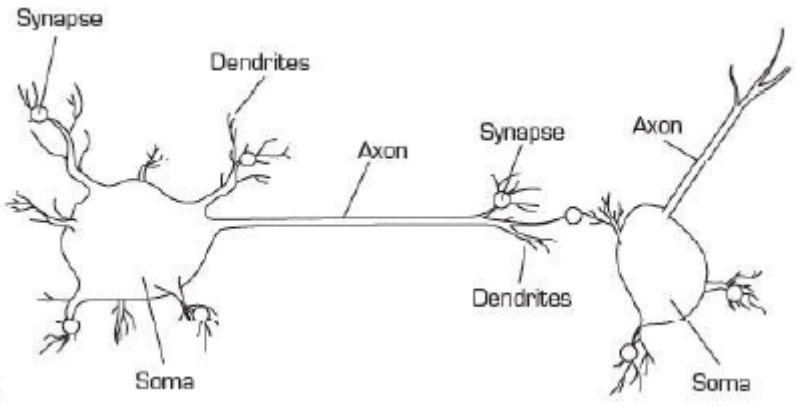

Fig. 2.1. Sinapsis neuronal

Este tipo de células pueden abarcar cinco funciones principales:

- Captar la información que reciben como impulsos procedentes de los órganos efectores u otras neuronas.
- Integrar dicha información en el código de activación propio de la neurona.
- Transmitir la información codificada en forma de impulsos a través de su axón.
- Distribuir mensajes.
- Enviar los impulsos a las neuronas siguientes a través de sus terminales.

2.2. Modelo computacional

En esta sección se presentan los conceptos más relevantes para entender una red de neuronas: neurona artificial, estructura de una red y aprendizaje.

2.2.1. La neurona artificial

Una neurona artificial es una unidad que recibe señales y posee un estado interno denominado nivel de activación. Dicho estado de activación puede ser cambiado por las señales recibidas. Además, para este cambio de nivel de activación, las neuronas artificiales poseen una función denominada de transición o función de activación.

El valor de activación de una neurona o célula depende de los valores (pesos) de las conexiones sinápticas y las entradas recibidas, pero en el caso de este trabajo, no existe dependencia con el valor de activación anterior. Para calcular dicho estado, primero se debe calcular la entrada total a la célula, E_i. Dicho valor resulta de la suma de todos los

valores de las entradas ponderados por el peso sináptico correspondiente.

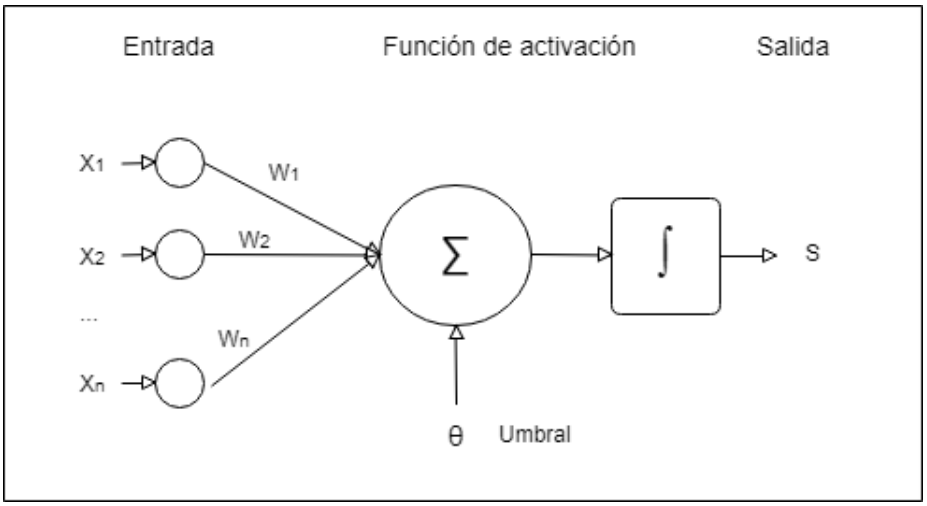

Fig. 2.2. Unidad de proceso

En la figura 2.2 se puede observar como un conjunto de entradas $\{x_1, x_2, \ldots, x_n\}$ se introducen en una neurona artificial. Dichas entradas, las cuales quedarán descritas por el vector X, corresponden a las señales de entrada a la neurona. Cada una de estas entradas se pondera por un valor asociado $\{w_1, w_2, \ldots, w_n\}$ y después se aplica el sumatorio. Cada peso indica la fuerza de la conexión sináptica correspondiente, y se representa con el vector W:

$$E = x_1 w_1 + x_2 w_2 + ... + x_n w_n + \theta = X^T W + \theta$$

Donde θ es el valor umbral.

A la expresión E, se le aplica una función de activación, F, la cual genera la salida de las neuronas. Las funciones de activación más comunes son:

- **Lineal**: $S = KE$. Generalmente $K = 1$
- **Umbral**: $S = 1$ si $E \geq \theta$, $S = 0$ si $E < \theta$; donde θ es un valor umbral constante.
- **Cualquier función**: $S = F(I)$ donde F corresponde a una función cualquiera.

2.2.2. Estructura básica de una red

En la figura 2.3 se puede observar la estructura más común de Red de Neuronas Artificial. En la parte izquierda se aprecia un conjunto de entradas, cada una de las cuales se conecta con la capa siguiente. Una vez se calcula la salida de la neurona, esta sirve como entrada para la neurona de la siguiente capa a la que está conectada.

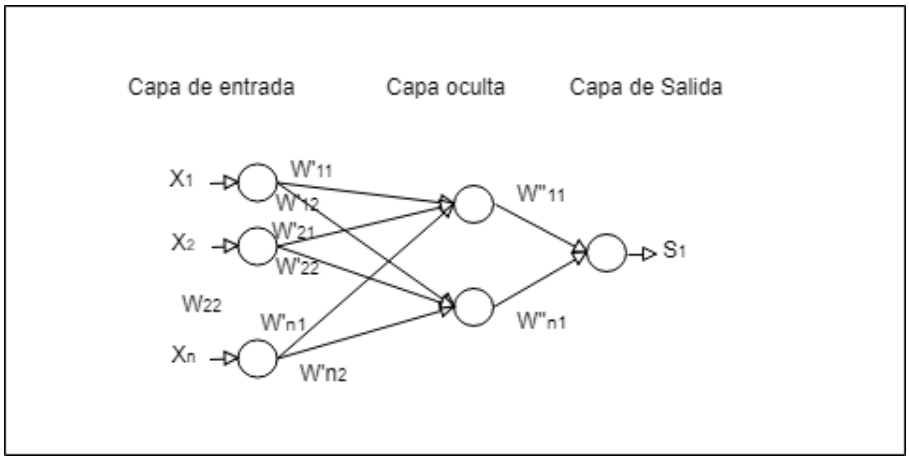

Fig. 2.3. Estructura típica de una red de neuronas artificial

En el primer nivel se encuentran las neuronas de entrada; éstas reciben los valores de los patrones que sirven como entrada a la red. A continuación se aprecia una serie de capas intermedias, llamadas ocultas, cuyas neuronas responden a rasgos particulares que pueden aparecer en los patrones. Puede haber una o muchas capas ocultas. Por último, se encuentra la capa de salida. Los valores producidos por las neuronas de dicha capa sirven como salida de la red.

El funcionamiento de la una red es el siguiente: se introduce cada patrón de entrada en la red, cada célula de la red, una vez recibidas todas los patrones, las procesa y genera una salida que se propaga a través de las conexiones, sirviendo de entrada a la neurona de la siguiente capa. Después de que la entrada haya sido propagada por toda la red se producirá un vector de salida, cuyos componentes son cada uno de los valores que han sido producidos por las neuronas de la última capa.

2.2.3. Aprendizaje

La parte más importante de una red de neuronas artificiales es el aprendizaje.

Las redes neuronales son modelos de aprendizaje basados en ejemplos, por lo que la capacidad que tenga la red para solucionar el problema tendrá una fuerte dependencia con el tipo de ejemplos que se tenga a la hora de construir el modelo. Con todo esto, un buen conjunto de entrenamiento debe tener las siguientes características:

- **Ser significativo**: Debe contener un número de ejemplos suficiente
- **Ser representativo**: Todas las clases o situaciones que deseen ser aprendidas o modeladas deben estar representadas dentro del conjunto de datos

El proceso de aprendizaje de una red de neuronas consiste en determinar los valores de los pesos para todas las conexiones con el objetivo de optimizar un determinado criterio o función). Dicho proceso se realiza mediante la introducción paulatina de todos los ejemplos del conjunto de entrenamiento modificando los pesos de las conexiones según un esquema de aprendizaje determinado. Una vez se han presentado todas las instancias de entrenamiento a la red, se comprueba si se ha cumplido el criterio de parada; si no es así se repite el proceso de nuevo.

El criterio de parada o convergencia depende del tipo de red utilizada, pero en términos generales, la finalización del proceso de entrenamiento se puede determinar mediante:

- Número fijo de ciclos.
- Cuando el error descienda por debajo de una cantidad establecida.
- Cuando la modificación de los pesos sea irrelevante.

Dependiendo del esquema de aprendizaje y del problema a resolver, se pueden distinguir diferentes tipos de esquemas de aprendizaje:

- **Aprendizaje supervisado**. En este tipo de aprendizaje, los datos del conjunto de entrenamiento tienen dos tipos de atributos: los datos en sí (entradas), y una información relativa a la solución del problema (valor de la clase o valor a aproximar, es decir, salida). Esta información se utilizará para modificar los pesos de las conexiones. Cada vez que se introduce un ejemplo y se genera la salida correspondiente, ésta se compara con la salida que se debería haber obtenido y la diferencia entre ambas, es decir, el error cometido por la red, determinará la manera en la que se cambiarán los pesos de las conexiones. Ver figura 2.4.

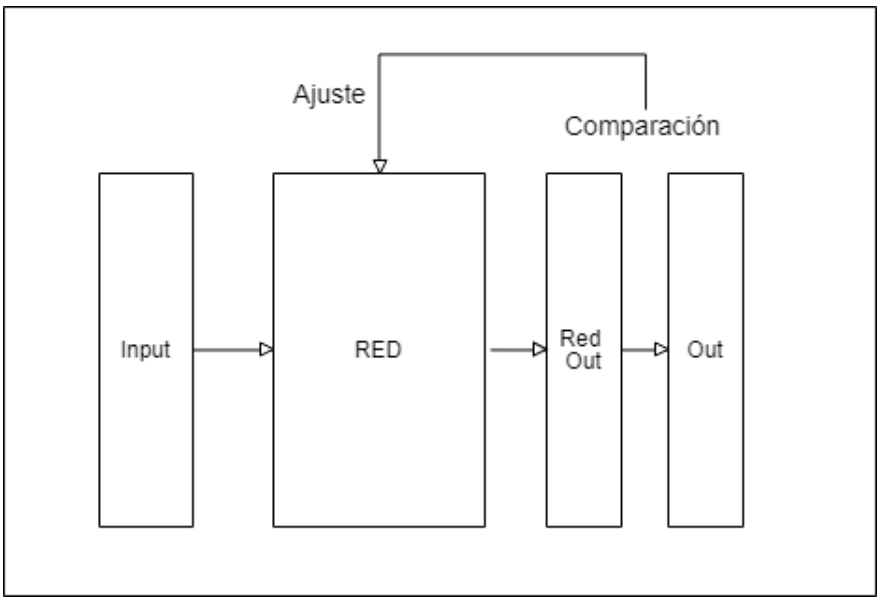

Fig. 2.4. Proceso de Aprendizaje Supervisado

Los modelos neuronales que más se utilizan con este tipo de aprendizaje son: Adaline, Perceptron Multicapa, Redes de de Base Radial y Redes Parcialmente recurrentes.

- **Aprendizaje no supervisado**.En este aprendizaje los datos del conjunto de entrenamiento sólo tienen información de las entradas del problema, no se conoce información relativa a la solución. La red modificará los pesos de las conexiones en función de las características comunes encontradas en los datos de entrenamiento. El proceso general que se sigue en este tipo de aprendizaje queda definido en la figura 2.5.

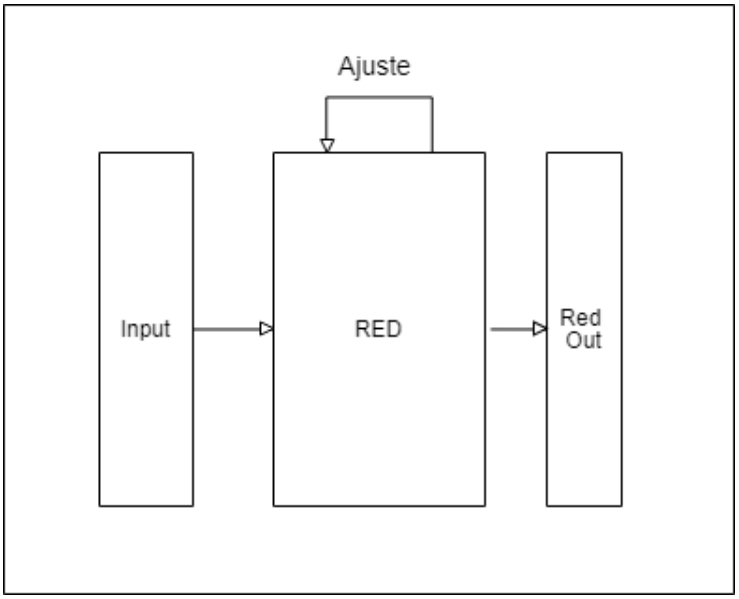

Fig. 2.5. Proceso de Aprendizaje No Supervisado

Los modelos neuronales con aprendizaje no supervisado más utilizados son: Mapas de Kohonen, ART, Red de Hopfield.

2.2.4. Capacidad de generalización

Una vez realizado el aprendizaje de una red neuronal, se deberá evaluar cómo se comporta dicha red ante la entrada de patrones distintos a los utilizados durante el proceso de aprendizaje. No sólo es importante saber si la red ha respondido bien ante los patrones de entrenamiento, sino que debe responder adecuadamente antes patrones no utilizados durante el aprendizaje. Este proceso es conocido como capacidad de generalización de la red. Para que una red obtenga una buena capacidad de generalización, es importante utilizar conjuntos de datos distintos en las fases de entrenamiento y test.

Por otro lado, es conveniente disponer de un conjunto de validación, que se suele utilizar para decidir los valores más adecuados de los hyper-parámetros que intervienen en el diseño y aprendizaje de las redes de neuronas, como se describe en la sección 4.2.4.

2.3. Perceptron Multicapa

Una de las redes de neuronas más simples es el Peceptron Simple (PS) y su regla de aprendizaje es conocida como Regla Delta. Está compuesto por una única neurona con función de activación umbral. Esta red fue diseñada para resolver problemas de clasificación lineal y sólo puede proporcionar soluciones lineales.

El Perceptron Multicapa o red multicapa con conexiones hacia delante es una generalización del PS y debe su origen a los problemas que presenta dicho modelo ante un problema que no fuera linealmente separable.

En 1969, Minsky y Papert, demostraron que la unión de varios Perceptrones podría resultar útil a la hora de abordar el problema de no linealidad. A pesar de ello, no se mostró una manera de ajustar los pesos de las conexiones entre la capa de entrada y la capa oculta y la regla delta utilizada para el aprendizaje del PS no podría ser utilizada. De todas formas, la idea de combinar varios Perceptrones fue la base para los estudios de Rumelhalt, Hinton y Willians en 1986, los cuales mostraron una forma de propagar hacia atrás (hacia la capa oculta) los errores cometidos en la salida de la red. Esta forma de calcular el error se denominará regla delta generalizada.

Diferentes autores han demostrado que el Perceptron Multicapa es un aproximador universal, puesto que con al menos una capa oculta, puede aproximar cualquier función continua sobre un compacto de R^n

2.3.1. Arquitectura del Perceptron Multicapa

El Perceptron Multicapa organiza su estructura en capas de neuronas como se muestra en la figura 2.6.

- **Capa de entrada**: se encarga de recibir las señales y los patrones de entrada.
- **Capa de salida**: proporciona la respuesta al exterior de los datos recibidos.
- **Capas ocultas**: procesan de forma no lineal los datos recibidos.

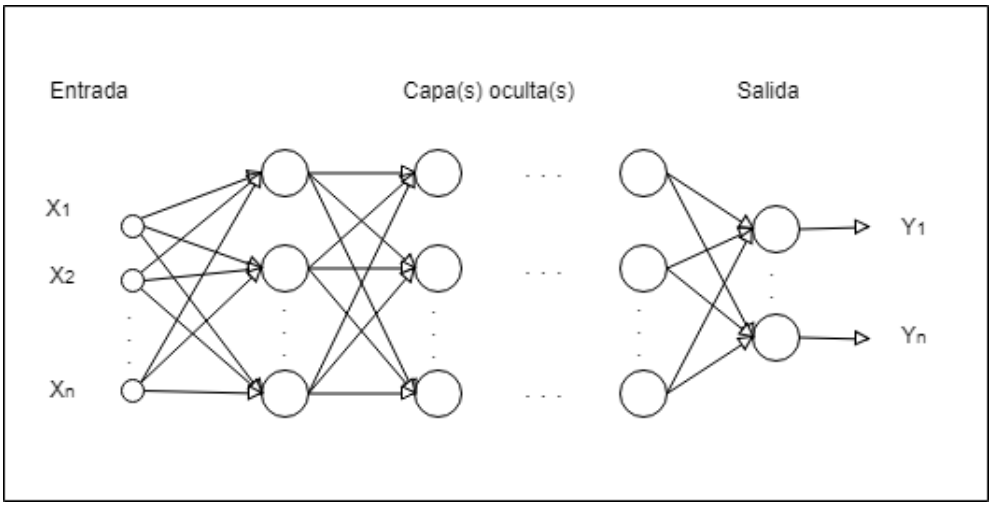

Fig. 2.6. Arquitectura del Perceptron Multicapa

Como se puede observar en la figura 2.6, las conexiones del Perceptron Multicapa se realizan únicamente de una capa a la siguiente. Este es el motivo por el que recibe el nombre de redes "feedfordward" o alimentadas hacia delante. Estas conexiones llevan asociado un peso, y a todas las neuronas se les asocia un umbral que suele tratarse como un peso más.

Por lo tanto, para un Perceptron Multicapa con c capas y nc neuronas en la capa $C = 1, 2, \ldots, n$ se tendrá una matriz de pesos W^c de la capa c a la $c + 1$ y un vector de umbrales U^c, dados por:

$$W^c = \begin{pmatrix} W^c_{11} & \ldots & W^c_{1n} \\ \ldots & \ldots & \ldots \\ W^c_{n1} & \ldots & W^c_{nc} \end{pmatrix}$$

$$U^c = \begin{pmatrix} U^c_1 \\ \ldots \\ U^c_n \end{pmatrix}$$

2.3.2. Propagación de los patrones de entrada

Mediante el Perceptron Multicapa se construye una relación entre las variables de salida y las de entrada. Dicha relación es obtenida por medio de la propagación hacia delante de los patrones de entrada. Este proceso se realiza mediante el procesamiento de dicha información por cada neurona de entrada, las cuales producen una señal de respuesta que será transferida a las neuronas de la siguiente capa a través de las conexiones correspondientes.

Sea un Perceptron Multicapa con un número de capas C, y n_c neuronas en la capa c. Sea $W^c = (w_{ij}^c)$ la matriz de pesos de la red ; y sea y $U^c = (u_i^c)$ el vector de umbrales. Denotamos la activación de la neurona i de la capa c como a_i^c. El cálculo de estas activaciones se realizará de la siguiente manera:

- **Capa de entrada**, a_i^1. En esta capa las neuronas sólo se encargan de traspasar las señales de entrada a la siguiente capa. Por lo que la activación será:

$$a_i^1 = x_i$$

- **Capas ocultas**, a_i^c. Las células de las capas ocultas procesan la información recibida aplicando la función de activación al sumatorio del producto de su entrada por el peso de la conexión. Es decir:

$$a_i^c = f(\sum_{j=1}^{n_{c-1}} w_{ji}^{c-1} a_j^{c-1} + u_j^{c-1})$$

  Donde a_j^{c-1} representa las activaciones de las neuronas de la capa anterior.

- **Capa de salida**, a_i^C. La activación de las neuronas de salida se calcula de igual modo que las activaciones de las capas ocultas:

$$y_i = a_i^C = f(\sum_{j=1}^{n_{C-1}} w_{ji}^{C-1} a_j^{C-1} + u_j^{C-1})$$

  Donde y_i es la salida de la neurona i de la última capa C

Las funciones de activación más comunes para el Perceptron Multicapa son la sigmoidal y la tangente hiperbólica, y de manera general, esta función de activación es la misma para todas las neuronas de la red, a excepción de las neuronas de salida que pueden tener función de activación lineal.

2.3.3. Algoritmo de retropropagación

Se llamará regla de aprendizaje al mecanismo seguido por la red para ajustar todos sus parámetros. Concretamente, en el caso del Perceptron Multicapa, se trata de un aprendizaje supervisado puesto que se conocen los valores de salidas deseados para los patrones de entrada. Sabiendo esto, los parámetros de la red se irán ajustando con el objetivo de minimizar la diferencia entre el valor calculado por la red y el valor deseado.

Dado que el objetivo es minimizar el error cometido por la red, es decir, que la salida de la red sea lo más parecida posible a la salida deseada, el proceso de aprendizaje se puede tomar como un problema de minimización con la siguiente forma:

$$Min_w e$$

Donde W es el conjunto de parámetros de la red (umbrales y pesos) y E la función de error que mide el error cometido por la red, el cual suele venir expresado por:

$$E = \frac{1}{N} \sum_{n=1}^{N} e(n)$$

Donde $e(n)$ es el error que comete la red para el patrón n:

$$e(n) = \frac{1}{2} \sum_{n=1}^{n_c} (s_i(n) - y_i(n)^2$$

La existencia de funciones de activación no lineales hace que el problema de minimización del error cometido por la red sea un problema no lineal, por lo que habrá que aplicar técnicas de optimización no lineales para encontrar la solución a este problema. Estas técnicas, normalmente se basan en un reajuste de los parámetros en una dirección de búsqueda. Concretamente, el Perceptron Multicapa utiliza la dirección negativa del gradiente de la función de error E. A este método se le denomina método del descenso de gradiente, y consiste en una minimización sucesiva de los errores cometidos en cada patrón.

Como las neuronas del Perceptron están agrupadas en capas con varios niveles es posible aplicar el método de descenso del gradiente de una manera eficiente. Esto se conoce como algoritmo de retropropagación, regla delta generalizada o propagación hacia atrás, y debe su nombre a que primero se calcula el error cometido en las neuronas de salida y este se va propagando hacia la capa de entrada.

2.3.4. Resumen de la Regla delta generalizada

Para modificar el valor de un peso de la red basta con tener en cuenta la activación de la neurona de la que parte la conexión y el término δ de la neurona a la que llega la conexión. Este término equivale al error que cada neurona propaga hacia las neuronas de la capa anterior. Dicho de otro modo, cada neurona de la capa oculta recibirá cierto valor de error (δ) de cada neurona de salida, y la suma de esos valores será el término δ que poseerá la neurona oculta. Con los valores δ de una capa se pueden calcular los valores δ de la capa anterior, y así sucesivamente hasta llegar a la capa de entrada.

El término δ de una neurona de salida se calcula mediante la derivada de la función de activación de esa neurona y su error. El cálculo del δ de cualquier otra neurona, se calculará utilizando la derivada de la función de activación de dicha neurona y la suma (ponderada por los pesos correspondientes) de los términos δ de las neuronas de la siguiente capa. Por tanto, cada neurona de salida envía su error (δ) hacia todas las neuronas de la capa anterior que se conectan a ella multiplicado por el peso de dicha conexión.

De esta manera, cada neurona oculta recibirá un cierto error δ de cada neurona de salida. Si se suman todos los errores recibidos se obtendrá el valor δ de la neurona. Estos valores se van propagando hacia atrás hasta la capa de entrada.

Las expresiones utilizadas para el cálculo de los términos δ son las siguientes:

Para las neuronas de la capa de salida:

$$\delta_j^C = (s_j - y_j) \cdot y_j(1 - y_j)$$

Para el resto de neuronas de la red:

$$\delta_j^c = a_j^{c+1}(1 - a_j^{c+1}) \sum_{i=1}^{n_{c+2}} \delta_i^{c+2} w_{ji}^{c+2}$$

Las expresiones para modificar los pesos y umbrales de la red son las siguientes:

Para la capa de salida:

- Pesos:

$$w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) + \alpha\delta_j^C(n)a_j^{C-1}(n)$$

- Umbrales:

$$u_j^C(n) = u_j^C(n-1) + \alpha\delta_j^C(n)$$

Para el resto de capas:

- Pesos:

$$w_{ki}^c(n) = w_{kj}^c(n-1) + \alpha\delta_j^{c+1}(n)a_k^c(n)$$

- Umbrales:

$$u_j^{c-1}(n) = u_j^{c-1}(n-1) + \alpha\delta_j^{c+1}(n)$$

2.3.5. Proceso de aprendizaje del PM

El algoritmo de aprendizaje del Perceptron Multicapa queda definido por los siguientes pasos:

1. Inicialización de los pesos y umbrales (bias) de la red con valores aleatorios cercanos a cero.
2. Se presenta a la red un patrón n se entrenamiento ($x(n)$, $s(n)$). Donde $x(n)$ son los n atributos que poseen los datos y $s(n)$ se corresponde con la salida deseada. La salida producida por las neuronas de entrada se propaga hacia la salida, obteniendo la repuesta de la red, $y(n)$.
3. Se evalúa el error cometido por la red, $e(n)$, para el patrón n.
4. Se aplica la regla delta generalizada para modificar los pesos y umbrales de la red. Para ello:
   a) Se calculan los valores δ para todas las neuronas en la capa de salida.
   b) Se calculan los valores δ para el resto de neuronas de la red, empezando desde la última capa oculta y retropropangando dichos valores hacia la capa de entrada.
   c) Se modifican los pesos y umbrales.
5. Repetir 2, 3 y 4 para todos los patrones de entrenamiento, completando así un ciclo de aprendizaje.
6. Se evalúa el error total cometido por la red. A dicho error se le denomina error de entrenamiento.
7. Se presentan los patrones de Test (o validación) calculando únicamente la salida de la red, sin modificar los pesos, y se evalúa el error total en el conjunto de Test (o validación).
8. Repetir 2, 3, 4, 5, 6 y 7 hasta alcanzar un mínimo del error de entrenamiento. Para ello se realizarán m ciclos de aprendizaje, aunque también se pueden utilizar otros criterios de parada:
   - Error estable de entrenamiento.
   - Error estable de validación.
   - El error de validación empieza a aumentar.

2.3.6. Hyper-parámetros del PM

Una vez se ha decidido abordar un problema utilizando un Perceptron Multicapa, una de las primeras decisiones que se deben tomar es el diseño de la arquitectura de la red. Para ello se deben determinar los siguientes parámetros:

- Función de activación a utilizar.
- Número de neuronas a utilizar.
- Número de capas de la red.

El número de neuronas en las capas de entrada y salida suele venir definido por el tipo de problema a resolver. En cuanto a la función de activación, la elección de una función u otra depende del recorrido que se quiera representar, pero no suele suponer una gran diferencia en la capacidad de generalización de la red.

Por otro lado, la topología de la red, número de capas ocultas y cantidad de neuronas en cada capa suele ser decidido por el diseñador. En el caso de este proyecto, tanto la topología de la red como el número de ciclos de entrenamiento se han decidido mediante la observación de los resultados de cada modelo después de un barrido de parámetros inicial y utilizando un conjunto de datos diferente al conjunto de entrenamiento y test (conjunto de validación). Este proceso de búsqueda será descrito en detalle en 4.2.4

Capítulo 3

Predicción de radiación global y directa

En este capítulo se describirán los datos utilizados para la realización del proyecto y el problema a resolver.

3.1. Descripción de los datos

Para el desarrollo del proyecto se ha utilizado un conjunto de datos medidos desde Marzo de 2015 hasta Marzo de 2017 en una estación de Sevilla, donde la radiación horizontal global (cantidad de radiación solar de onda corta recibida en una superficie horizontal) ha sido recogida con un Kipp & Zonen CMP6 con un intervalo de quince minutos de diferencia entre muestra y muestra. Los datos han sido proporcionados por el grupo MATRAS de la Universidad de Jaén (www.uja.es)

El conjunto de datos inicial está formado por 1754400 observaciones.

```
[1] "date"           "horizon"        "dniKt_expected" "ghiKt_expected"
[5] "zenit"
```

Fig. 3.1. Variables utilizadas en el proyecto

En cuanto a la nomenclatura de las variables mostradas en la figura 3.1 hay que tener las siguientes consideraciones:

- Los atributos definidos como **Kt** son los valores de la radiación normalizada por la radiación de cielo despejado. Se usa esta medida de radiación para evitar tener que aproximar el ciclo diario del sol.
- Las variables denominadas como **_expected** se refieren a la variable en cuestión medida en el horizonte de predicción. Es decir si se está en t y se quiere predecir en

$t + 15$ minutos, **_present** se corresponde con el valor de la variable en t y _expected en $t + 15$

Teniendo esto en cuenta se pasa a definir las variables utilizadas en el proyecto:

- **Date**: fecha y hora en formato Posixct de la medición de los valores de radiación. Las observaciones de este atributo irán desde el 2015-03-01 00:00:00 hasta el 2017-02-28 23:45:00 con un muestreo de quince minutos.
- **Horizon**: Horizonte temporal de predicción. En total hay 25 horizontes temporales, desde el momento actual (horizon = 0) hasta 6 horas en el futuro (horizon = 360). El valor del horizonte irá aumentando de quince en quince. Es decir, en el momento actual nos encontramos en el horizonte 0, en caso de querer predecir quince minutos en el futuro, nos encontraremos en el horizonte 15, y así sucesivamente.
- **Ghikt_expected**: Medición normalizada de la radiación horizontal global. Kt es el ratio entre GHI y la radiación global con el cielo despejado GHI_{clear}
- **DniKt_expected**: Medición normalizada de la radiación difusa normal a la superficie.
- **Zenit**: Punto de la bóveda celeste en el que se encuentra el sol. Medido en grados. La variable zenit será utilizada para el filtrado de los datos, puesto que, para un $zenit \geq 75$ la medida de radiación es nula.

Habiendo fijado las variables que se utilizarán en el proyecto, cabe comentar que un 71.3221 % de los datos tienen un valor nulos, por lo que se trabajará con un total de 503125 instancias.

Además de los datos mencionados anteriormente se dispone también de las predicciones realizadas por otros modelos, los cuales serán utilizados para comparar los modelos desarrollados en entre trabajo. Éstos son SmartPersistence y Satélite, que se describen a continuación:

- **Smart Persistence**. Este modelo calcula el valor de la radiación en un determinado instante de tiempo t con el valor de la radiación medido en el instante correspondiente, I_0, y se corrige con la radiación que se obtendría en el caso de que el cielo estuviera despejado, I_{cs}, desde el instante inicial t_0 hasta el momento que se quiera predecir, t [13]. La relación entre la radiación actual y la radiación con el cielo despejado en un momento determinado se mantiene constante y multiplica al valor del cielo despejado en un instante futuro t:

$$I(t) = \frac{I_0}{I_{cs}(0)} * I_{cs}(t)$$

- **Satellite**. En este modelo, las imágenes tomadas por un satélite son procesadas para generar lo que se denomina índice de imágenes. Después se realiza una comparación estadística de varios índices de imágenes consecutivos, lo que permite obtener el vector de movimiento de las nubes. Dicho vector discreto de movimiento, será transformado en un flujo continuo calculando las líneas de corriente como puede ser una familia de curvas tangente al viento. Estas líneas de corriente se utilizarán para obtener futuras clasificaciones de nubes. Posteriormente el valor del cielo despejado, y finalmente la predicción de GHI [13].

Las predicciones proporcionadas por estos modelos se utilizarán con el propósito de comparar y evaluar las predicciones proporcionadas los modelos estadísticos construidos.

En la tabla 3.1 se muestra un ejemplo del aspecto de los datos utilizados para la generación de las ventanas de predicción:

Instancia	date	horizon	ghiKt_expected	dniKt_expected	zenit
1	2015-03-02 13:15:00	0	0.93413162	0.9096509	45.32247
2	2015-03-02 13:30:00	0	0.92433334	0.8931156	46.33007
3	2015-03-02 13:45:00	0	0.93869638	0.9117936	47.57368
4	2015-03-02 14:00:00	0	0.93850018	0.9145342	49.03484
5	2015-03-02 14:15:00	0	0.93656798	0.9145380	50.69413
6	2015-03-02 14:30:00	0	0.89427079	0.8760862	52.53211

Tabla 3.1. Muestra inicial de los datos

3.1.1. Descripción del problema

El problema al que nos enfrentamos, la predicción de la radiación global y directa en diferentes horizontes temporales, es un problema de predicción de series temporales. Es decir, a partir de una colección de valores pertenecientes a un suceso a lo largo del tiempo $x(0), x(1), \ldots, x(t)$ se pretende averiguar el comportamiento que tendrá dicho suceso en un instante futuro $x(t + h)$, siendo h el horizonte de predicción.

La evolución en el tiempo de un suceso puede no depender de la variable tiempo sino que el comportamiento de la serie puede venir determinado por otros factores como pueden ser los valores de la serie en instantes anteriores u otras variables que sí tengan cierta

dependencia temporal.

De forma habitual, únicamente se dispone de las observaciones de la serie, y es necesario interpretar dichas observaciones para poder generar algún modelo. Como se ha comentado anteriormente, se utilizarán modelos neuronales para estudiar estas dependencias, así como un modelo de regresión lineal.

En el ámbito de la predicción de series temporales se pueden utilizar varios tipos de redes de neuronas artificiales, ya sean modelos estáticos como puede ser el Preceptron Multicapa o las Redes de Neuronas de Base Radial, o modelos dinámicos, como son las redes recurrentes. Debido a que el Perceptron Multicapa está considerado como un aproximador universal y a que se pueden construir modelos de forma sencilla, será el tipo de red neuronal escogido para la realización del proyecto.

Dentro de las series temporales se suelen distinguir dos tipos de problemas: problemas de predicción a un paso de tiempo (predicción del siguiente instante) y problemas de predicción en diferentes horizontes temporales de predicción. El segundo engloba al primero. En este trabajo se abordará la predicción en diferentes horizontes temporales.

Cuando se aborda un problema de predicción en diferentes pasos de tiempo, se pretende averiguar el valor que tomará la serie en un instante $t + h$, con $h \geq 1$, dado la información conocida hasta el instante t, como se muestra en la figura 3.2.

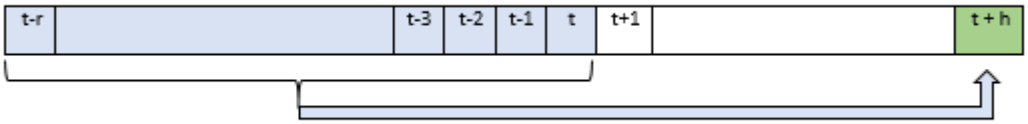

Fig. 3.2. Horizonte temporal de Predicción

3.1.2. Planteamiento del problema de predicción

Para abordar el problema, en este trabajo se construyen modelos para predecir directamente el valor de la serie en t+h, es decir $\tilde{x}(t + h)$

Para ello será necesario construir tantos modelos como horizontes a predecir para estimar el valor de $\tilde{x}$ a partir de la información disponible en x(t) (ver figura 3.3).

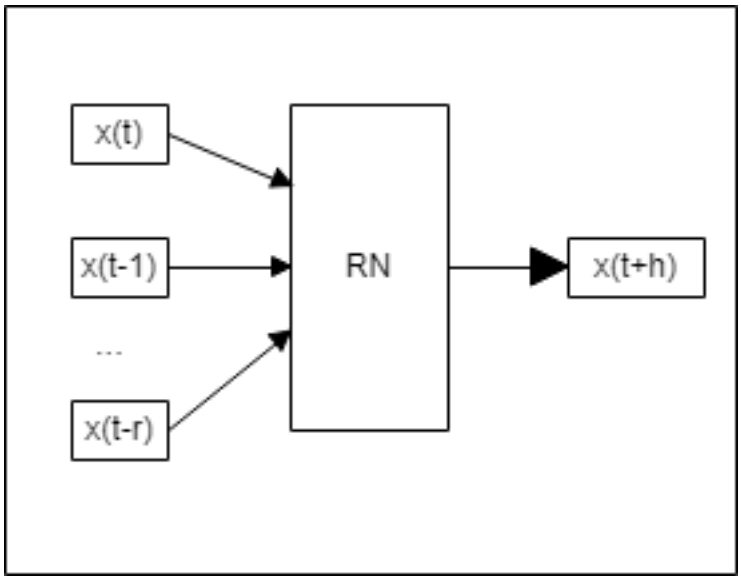

Fig. 3.3. Esquema de Predicción

Dado un horizonte temporal h, el modelo para predecir en dicho horizonte temporal viene dado por:

$$\tilde{x}(t+h) = F_h(x(t), (t-1), \ldots, x(t-r))$$

Siendo r el número de valores anteriores a utilizar y F_h la aproximación que se construirá con el PM o con un modelo de regresión lineal para el horizonte h. Como se ha comentado en Capítulo 1, se van a utilizar 24 horizontes de predicción, por lo que se construyen 24 modelos. La función F_h será aproximada con un PM y regresión lineal.

Los patrones necesarios para construir los modelos, es decir, el conjunto de datos de los cuales se extraerán los ficheros de entrenamiento, validación y test utilizados por la red neuronal, se construyen a través de una ventana deslizante como se muestra en la tabla 3.2

Patrón	**ENTRADA**	**SALIDA DESEADA**
Patrón 1	$x(r), x(r-1), \ldots, x(1), x(0)$	$x(r+h)$
Patrón 2	$x(r+1), x(r), x(r-1), \ldots, x(1)$	$x(d+h+1)$
Patrón 3	$x(r+2), x(r+1), x(r), \ldots, x(2)$	$x(d+h+2)$
...	...	...
Patrón (N-r-h)	$x(N-h-1), x(N-h-2), \ldots, x(N-h-r)$	$x(N)$

Tabla 3.2. Construcción de patrones para el horizonte de predicción h

Para abordar el problema es necesario determinar el valor de r a utilizar, es decir, el número de instantes anteriores necesario para conseguir una estimación que minimice el error producido por la red. Este tema se trata en la siguiente sección.

3.1.3. Elección de los r instantes anteriores

Al tratarse de una serie temporal seguramente haya cierta relación entre las variables con respecto del tiempo. Es decir, el valor observado en un instante $x(t-r)$ guardará cierta relación con la observación de la misma variable en $x(t)$. A este proceso se le denomina autocorrelación o correlación serial.

Existen varias formas de medir la dependencia de una variable consigo misma: [15]

- **Función de autocorrelación (acf)**. Mide la correlación entre dos variables en instantes distintos. Es decir, separadas k periodos.

$$\rho_j = corr(X_j, X_{j-k}) = \frac{cov(X_j, X_{j-k})}{\sqrt{V(X_j)}\sqrt{V(X_{j-k}}}$$

  Donde $corr(X_j, X_{j-k})$ es la relación existente entre una variable X en un instante j y la misma variable en un instante anterior $j-k$. $cov(X_j, X_{j-k})$ representa la covarianza de dicha variable en esos instantes de tiempo, y $V(X_j)$ es la varianza de la variable X en el instante j, por lo que $\sqrt{V(X_j)}$ indica la desviación estandar de la variable en j

  La función de autocorrelación posee las siguiente propiedades:

  - $\rho_0 = 1$
  - $-1 \leq \rho_j \leq 1$
  - Simetría: $\rho_j = \rho_{-j}$

- **Función de autocorreclación parcial (pacf)**. Mide la correlación entre dos variables separadas por k periodos sin considerar la dependencia entre ellas.

$$\prod_j = corr(X_j, X_{j-k}/X_{j-1}, X_{j-2}, ..., X_{j-k+1})$$

$$\prod_j = \frac{cov(X_j - \widehat{X}_{j-k}, X_{j-k} - \widehat{X}_{j-k})}{\sqrt{V(X_j - \widehat{X}_j)}\sqrt{V(X_{j-k} - \widehat{X}_{j-k}}}$$

Donde $cov(X_j, \widehat{X}_{j-k})$ es la relación existente entre una variable X en un instante j y la variable predicha en un instante anterior $j-k$. $V(\widehat{X}_j)$ es la varianza del valor predicho de la variable X en el instante j, por lo que $\sqrt{V(X_{j-k} - \widehat{X}_{j-k}}$ indica la desviación producida entre el valor real y el predicho.

- **Prueba de Ljung-Box**. Mediante esta prueba se puede determinar si los valores de una variable en instantes desplazados son independientes, esto ocurre si todos los coeficientes de autocorrelación son iguales a cero.

$$LB = n(n+2) = \sum_{k=1}^{m} (\frac{\widehat{\rho_k^2}}{n-k}) \sim X^2(m)$$

Donde n indica el tamaño del conjunto, m el retraso, y ρ_k indica la autocorrelación con la variable X en k

Para estimar el número de instantes anteriores en el problema que se trata y poder así construir las tablas de patrones entada-salida explicadas en el apartado anterior se utilizará la función acf de autocorrelación que proporciona el lenguaje de programación R sobre las variables ghiKt_expected y dniKt_expected con un retardo máximo de 40 instantes.

A diferencia de acf, la autocorrelación parcial (pacf) únicamente tiene en cuenta los valores X_t y X_{t+h} dejando a un lado las contribuciones desde X_{t+1} a X_{t+h-1}. Este es el motivo por el que se eligió acf para estudiar qué número de instantes coger para realizar las tablas de predicción.

Capítulo 4

Validación experimental

En este capítulo se presentará la validación experimental realizada en el proyecto. Este tema cubrirá el software y librerías utilizadas, la metodología empleada para realizar los experimentos y los resultados obtenidos por los mismos.

4.1. Software

R es un lenguaje de programación estadístico utilizado para el análisis de datos y la generación de gráficos, creado por Ross Ihaka y Roblert Gentelman. Este lenguaje se distribuye de manera gratuita bajo los términos de licencia pública general de GNU (www.gnu.org) y está disponible en distintos formatos, ya sea como código fuente o como archivos binarios precompilados.

En cuanto al entorno de desarrollo en el que se ha programado en R, se ha utilizado RStudio (www.rstudio.com). Esta aplicación es un entorno de desarrollo integrado, IDE, el cual integra las herramientas que se necesitan a la hora de desarrollar en el lenguaje: ayuda y documentación, manejo sencillo de directorios y proyectos, visor de gráficas, etc.

A la hora del desarrollo ha sido necesario utilizar algunas funciones contenidas en las librerías externas expuestas a continuación:

- **Lubridate** [16]. Paquete contenido en el repositorio CRAN en su versión 1.7.4 utilizado para tratar las fechas de una manera más sencilla. Las funciones que se ha utilizado de esta librería han sido:
    - as.POSIXct(x, origin = "1970-01-01", tz = , ...): Función utilizada para manipular las fechas en formato POSIXlt y POSIXct contenidas en los datos del problema. Según la documentación de la propia librería, el origen de la fecha

ha sido prefijado a "1970-01-01":

"Note: [. . .] the origin of time for the "POSIXctçlass, '1970-01-01 00:00.00 UTC"

- month(x): Función que toma/modifica el componente de una fecha correspondiente al mes del año.
- day(x): Función que toma/modifica el componente de una fecha correspondiente al día del mes.

- **Data.table**[17]. Paquete contenido en el repositorio CRAN en su versión 1.11.4 utilizado para la agregación y manejo de grandes conjuntos de datos de una manera sencilla.

- **Dplyr** [18]. Paquete contenido en el repositorio CRAN en su versión 0.7.5. Es un potente paquete de R para la transformación y extracción de características de conjuntos de datos con forma tabular.

  - Select(.data,. . .): Función utilizada para la selección de columnas de una tabla.
  - Filter(.data,. . .): Función utilizada para la selección de filas de una tabla.
  - Arrange(.data,. . .). Función utilizada para la reordenación de las filas de una tabla en base de una columna determinada.
  - Pipe: %> %. Operador que se utiliza para redirigir la salida de una función a la entrada de otra en lugar de tener que anidar ambas funciones.

- **RSNNS** [19]. Paquete contenido en el repositorio CRAN en su versión 0.4-10. Este paquete contiene la implementación de varios modelos de redes neuronales. En este trabajo se ha utilizado el modelo de preceptron multicapa de la librería.

  - mlp(x, y, size = c(5), maxit = 100, learnFunc = "Std_Backpropagation", learnFuncParams = c(0.2,. . .), ...): Función que crea un percetron multicapa con los parámetros especificados y lo entrena.
  - lm(formula, data, subset, weights, na.action,method = "qr", model = TRUE, ...): Funcion utilizada para entrenar los modelos lineales.

- **Ggplot2** [20]. Paquete contenido en el repositorio CRAN en su versión 2.2.1 utilizado a la hora de generar los gráficos comparativos.

4.2. Metodología empleada

A continuación se presentará la metodología seguida durante la construcción de los modelos de predicción en los distintos horizontes temporales. El trabajo realizado en esta parte del proyecto se compone de tres fases: Preparación de los datos y las ventanas de predicción, generación de los modelos para la predicción de la radiación global (GHI) y la radiación directa (DNI) y evaluación de los errores de los mismos, y comparación de los resultados producidos por las redes neuronales con los modelos disponibles y proporcionados por el grupo MATRAS de la Universidad de Jaén (SmartPesistence y Satélite). Se presentará también la comparación con modelos de regresión lineal.

4.2.1. Preparación de los datos

Antes de empezar a generar los modelos de redes neuronales que servirán como predictores de las medidas de radiación en los distintos horizontes temporales, es necesario realizar un preprocesado de los datos para generar los ficheros de entrenamiento, validación y test que se utilizarán.

Como se ha comentado en el Capítulo 3, se utilizarán técnicas de predicción de series temporales en múltiples pasos de tiempo. Para ello, se realizó, en primer lugar, un estudio de la autocorrelación existente en las variables a predecir (GHI y DNI) con el objetivo de determinar el número de valores anteriores a utilizar para la construcción de los modelos y, por tanto, para la preparación de los datos.

El resultado de dicho estudio se puede observar en la siguiente figura 4.1:

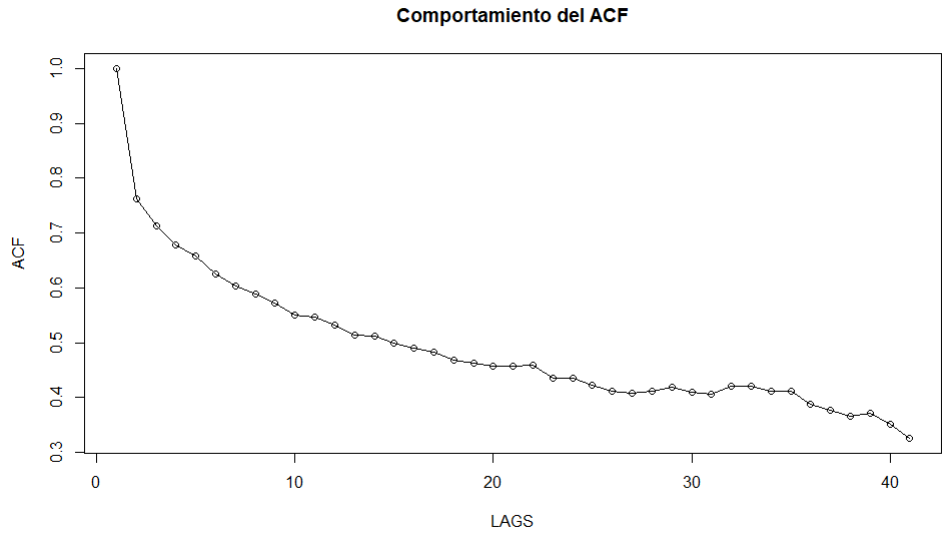

Fig. 4.1. Comportamiento acf

Es importante mencionar que los resultados del estudio del comportamiento del acf para ambas variables, GHIkt y DNIkt, fueron iguales.

Una vez se tengan los resultados de la función, se tomará como límite un valor de autocorrelación de 0.5, con lo que se elegirá un r = 15, para todos los horizontes de predicción. Como el muestreo de los datos es cada 15 minutos, elegir r=15 indica que los modelos construidos utilizarán información de la serie temporal correspondiente a 15*15 minutos.

Después de fijar el número de instantes anteriores a coger, se generan tablas para cada uno de los horizontes de predicción siguiendo el esquema de ventana deslizante especificado en la sección 3.1.2 (tabla 3.2), obteniendo así las tablas (ver Tabla 4.2) de las que se obtendrán los patrones para construir los modelos.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{t+h}
0,38	0,4	0,39	0,38	0,39	0,39	0,39	0,33	0,41	0,35	0,4	0,44	0,53	0,52	0,72	0,93
0,4	0,39	0,38	0,39	0,39	0,39	0,33	0,41	0,35	0,4	0,44	0,53	0,52	0,72	0,93	0,931
...	...														...

Fig. 4.2. Patrones para el horizonte h

Para finalmente generar los patrones, es necesario, en primer lugar, aplicar un filtro para eliminar los datos relativos a la noche y, en segundo lugar, eliminar o procesar los datos faltantes.

Filtro por Zenit. Cuando el zenit toma un valor mayor de 75, quiere decir que es de noche, con lo que las mediciones de radiación solar serán cero. Se eliminan entonces las filas para las cuales se tiene $Zenit(t + h) \geq 75$

Tratamiento de valores ausentes. Como se ha comentado en la sección 3.1, el conjunto posee un 71.3221 % de datos faltantes. Se ha tomado la decisión de eliminar la fila si existe en ella algún NA.

Es importante realizar este proceso de filtrado en este orden, ya que, si se filtra por el zenit o NA antes de generar las ventanas deslizantes, se perdería el carácter temporal de la serie.

Después de filtrar los datos por zenit y eliminar los valores ausentes, el número de instancias disponibles para cada horizonte de predicción es el que se muestra en la tabla 4.1.

Horizonte	Observaciones	Horizonte	Observaciones	Horizonte	Observaciones
15	8165	135	5240	255	2467
30	7794	150	4811	270	2180
45	7460	165	4429	285	19081
60	7145	180	4061	300	1634
75	6794	195	3718	315	1371
90	6418	210	3380	330	1154
105	6047	225	3062	345	966
120	5643	240	2757	360	810

Tabla 4.1. Tabla. Número de Instancias por horizonte.

4.2.2. Conjuntos de entrenamiento, validación y test

Una vez filtrados los datos, a partir de las tablas de entrada-salida (tablas 4.2) se generarán para cada horizonte de predicción los datos de entrenamiento, validación y test, se realizará la separacón en conjuntos de entrenamiento, validación y test. Cuyas funciones son:

- El conjunto de Entenamiento se utiliza para entrenar los modelos.
- El conjunto de Validación sirve para decidir los mejores hyper-parámetros
- El conjunto de Test es utilizado para estudiar el rendimiento de los modelos

Es importante destacar que para el problema que se aborda en el trabajo y para estudiar la operacionalidad de los modelos, la separación de los conjuntos de entrenamiento, validación y test se realiza utilizando semanas completas.

En este trabajo y con el objetivo de validar ampliamente los modelos construidos, de manera que su capacidad se analice en dos semanas diferentes de test, se eligen dos criterios distintos para la separación de los datos (en entrenamiento - validación – test):

Criterio 1:

- Las dos primeras semanas de cada mes para entrenamiento
- La tercera semana de cada mes para validación
- La última semana de cada mes para test.

Criterio 2:

- Las dos primeras semanas de cada mes para entrenamiento
- La última semana de cada mes para validación
- La tercera semana de cada mes para test.

4.2.3. Errores a evaluar

Para medir la calidad de predicción de los modelos, se mide la diferencia que se produce entre la salida de la red y el valor real de radiación medido en ese instante. En el contexto de radiación solar se suelen utilizar diferentes medidas de error:

- R^2, también denominado coeficiente de determinación, es una medida estadística del error la cual determina la calidad de un modelo y se define como la proporción de la varianza total de la variable tomada.

$$R^2 = 1 - \frac{\sum_{t=1}^{T}(\widehat{Y_t} - \overline{Y})^2}{\sum_{t=1}^{T}(Y_t - \overline{Y})^2}$$

  Esta medida puede tomar valores dentro del rango $0 \leq R^2 \leq 1$. Cuanto más cercano esté el error de la unidad, mejor será el modelo.

- RMSE o Root mean square error es una medida de error utilizada para medir la diferencia entre valores predichos y reales. Queda definido mediante la siguiente ecuación:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(O_t - P_t)^2}{N}}$$

  Donde O_t y P_t representan los valores observados y predichos en el instante t respectivamente y N la longitud del conjunto tomado.

- nRMSE, se define como el error RMSE normalizado. Para ello bastaría con dividir al RMSE por la media de los valores observados.

$$nRMSE = \frac{RMSE}{\overline{P}}$$

- MAE o mean absolute error, mide el error medio producido en las predicciones de una serie temporal. Se puede considerar como la aplicación de la media sobre la diferencia entre la salida observada y la predicha, y se definirá mediante la siguiente expresión:

$$MAE = \frac{\sum_{t=1}^{T} |O_t - P_t|}{N}$$

  Donde O_t y P_t representan los valores observados y predichos en el instante t respectivamente y N la longitud del conjunto tomado.

- nMAE, se define como el error MAE normalizado. Para ello bastaría con dividirlo por la media de los valores observados.

$$nMAE = \frac{MAE}{\overline{P}}$$

En este trabajo se implementaron inicialmente estas medidas, pero se decidió trabajar con RMSE y MAE con la idea de simplificar los resultados que se muestran. La razón por la que se eligen ambos errores (RMSE y MAE) es porque presentan comportamientos diferentes y en el contexto de radiación solar ambos errores son generalmente utilizados.

4.2.4. Búsqueda de los mejores hyper-parámetros y generación de modelos

Una vez separado el conjunto de datos inicial en los respectivos ficheros de entrenamiento, validación y test, el siguiente paso es determinar la mejor combinación de parámetros para los modelos de redes de neuronas construidos en este trabajo.

Para ello, se realizará un barrido de experimentos para cada horizonte combinando los siguientes parámetros de la red:

- Número de neuronas ocultas: 5, 10, 15, 20, 25, 30, 40, 50
- Valores de razón de aprendizaje: 0.001, 0.005, 0.01, 0.05, 0.1,0.2
- Todas las redes se entrenan 7000 ciclos y utilizando el error sobre el conjunto de validación, para cada red se elegirá el número de ciclos más apropiado.

Para realizar la búsqueda de hyper-parámetros, para cada horizonte, con cada una de las topologías, se entrenará la red con cada una de las razones de aprendizaje especificadas. En cuanto al número de ciclos utilizados, inicialmente para cada conjunto (T, R_A) se entrenará el modelo con un número máximo de ciclos, 7000.

Una vez se tiene el modelo entrenado con los 7000 ciclos, se busca el ciclo para el que la red comete el menor error de validación. Cuando se conozca dicho punto de mínimo error, se construye modelo final con el mismo conjunto (T, R_A), pero esta vez el proceso se realizará para el número de ciclos en ese punto.

A continuación se calcula también el error del modelo sobre el conjunto de test. De todas las redes construidas, se cogerá la red que menor error de validación tenga para una posterior comparación de los resultados con otros modelos.

El algoritmo general seguido para la búsqueda de los hyper-parámetros se presenta en al siguiente figura 4.3.

1.- ∀ *h en horizontes*:
2.- Carga de los ficheros de entrenamiento, validación y test
3.- ∀ *t en topologías*:
4.- ∀ *ra en razones de aprendizaje*:
5.- modelo <- mlp(x= train, y =taget, t, ra, ciclos = 7000)
6.- c <- min(modelo.IterativeTestError)
7.- Modelo <- mlp(x= train, y =taget, t, ra, ciclos = C)
8.- Errores <- Calculo de errores para el modelo:
- MAE(modelo)
- RMSE(modelo)

9.- Coger el modelo con menor error de validación

Fig. 4.3. Proceso para la búsqueda de Hyper-parámetros

Este proceso de experimentación se realizará para cada una de las terna entrenamiento/validación/test utilizadas en este trabajo (ver 4.2.1)

En lo que respecta a los modelos lineales, no es necesario hacer búsqueda de hiperparámetros, por lo que en este caso basta construir un modelo para cada horizonte utilizando el conjunto de entrenamiento y evaluar los errores, como se muestra en la figura 4.4

Para ello, para cada horizonte, se carga el fichero entrenamiento correspondiente, se genera el modelo de regresión lineal y se calculan los errores cometidos por el mismo. En este caso únicamente hay un modelo por horizonte, por lo que nos será necesario elegir el mejor modelo.

1.- $\forall$ *h en horizontes*:
2.- Carga de los ficheros de entrenamiento, validación y test
3.- modelo <- lm(SALIDA, trainSet)
4.- Errores <- Calculo de errores para el modelo:
    - MAE(modelo)
    - RMSE(modelo)
5.- No se elige ningún modelo en este caso, solo hay 1 para cada horizonte

Fig. 4.4. Generación de modelos lineales

4.2.5. Comparación con los modelos disponibles

Como se ha comentado en 4.2.4, los resultados de los modelos construidos en este proyecto se van a comparar con las predicciones proporcionadas por dos modelos, Smart-Persistence y Satélite, descritos en 3.1

El objetivo de esta comparación es medir la capacidad de generalización que posee el Perceptron Multicapa frente a modelos físicos que se utilizan hoy en día en el dominio de la predicción de radiación solar.

Para realizar dicha comparación, se dispone de las predicciones de GHI y DNI de estos modelos, pero será necesario calcular los errores cometidos por dichos modelos en las fechas de test correspondientes, estas fechas variarán en función de si se está tomando como conjunto de test la última o la tercera semana del mes. Para ello es necesario procesar las predicciones disponibles y escoger las coincidentes en fechas con los datos de test utilizados para construir los modelos neuronales y lineales. Puede darse el caso en que para las fechas de test correspondientes, los valores de los modelos disponibles (SmartPersistence y Satélite) sean nulos. En ese caso la fila entera queda descartada.

Para los modelos físicos en la cuarta semana se ha encontrado un 54.4 % de valores faltantes de media, mientras que para la tercera semana se tiene un 57.77 % de ellos.

El algoritmo general utilizado en este proceso de comparación queda definido en la figura 4.5.

1.- Carga de patrones de test
2.- Carga de las predicciones de los modelos disponibles (SmartPersistence y Satélite)
3.- Elección de las predicciones en las fechas coincidentes con el conjunto de test
4.- Eliminación de valores nulos
5.- Cálculo de los errores

Fig. 4.5. Generación de modelos lineales

4.3. Resultados experimentales

En este apartado se presentarán los resultados obtenidos de la experimentación previamente explicada. Para ello se presentarán distintas gráficas que recogen la evolución de los errores de los modelos en cada horizonte de predicción para las dos variables a predecir, GHI y DNI.

4.3.1. Resultados GHI

En las tablas 4.2 y 4.3 se presentan las configuraciones de hyper-parámetros de las redes que obtienen el menor error MAE en validación para GHI en cada horizonte temporal.

Horizonte	Topologia	RA	Ciclos
15	30	0,001	1704
30	50	0,001	1947
45	40	0,001	7000
60	10	0,01	6257
75	10	0,01	7000
90	10	0,01	7000
105	10	0,01	7000
120	10	0,01	7000
135	10	0,01	7000
150	10	0,01	6777
165	10	0,01	5165
180	15	0,05	1608
195	10	0,1	354
210	15	0,1	161
225	40	0,1	17
240	30	0,2	15
255	50	0,1	21
270	15	0,1	320
285	10	0,1	293
300	10	0,1	645
315	50	0,2	12
330	50	0,2	12
345	50	0,2	15
360	5	0,01	4720

Tabla 4.2. Mejores modelos de GHI para la cuarta semana de Test

Horizonte	Topologia	RA	Ciclos
15	30	0,001	3213
30	50	0,001	3994
45	50	0,001	6993
60	20	0,05	2387
75	10	0,01	7000
90	10	0,01	6999
105	10	0,01	7000
120	10	0,01	7000
135	10	0,01	7000
150	10	0,01	6176
165	50	0,01	7000
180	50	0,01	7000
195	40	0,05	1720
210	50	0,05	133
225	5	0,1	70
240	30	0,01	5474
255	15	0,01	7000
270	20	0,05	2035
285	10	0,1	874
300	15	0,1	1068
315	15	0,1	1530
330	15	0,1	2285
345	15	0,05	3733
360	25	0,001	1

Tabla 4.3. Mejores modelos de GHI para la tercera semana de Test

Resultados para la cuarta semana de Test

En este apartado, se procederá a presentar los resultados obtenidos por cada uno de los modelos para la cuarta semana de Test.

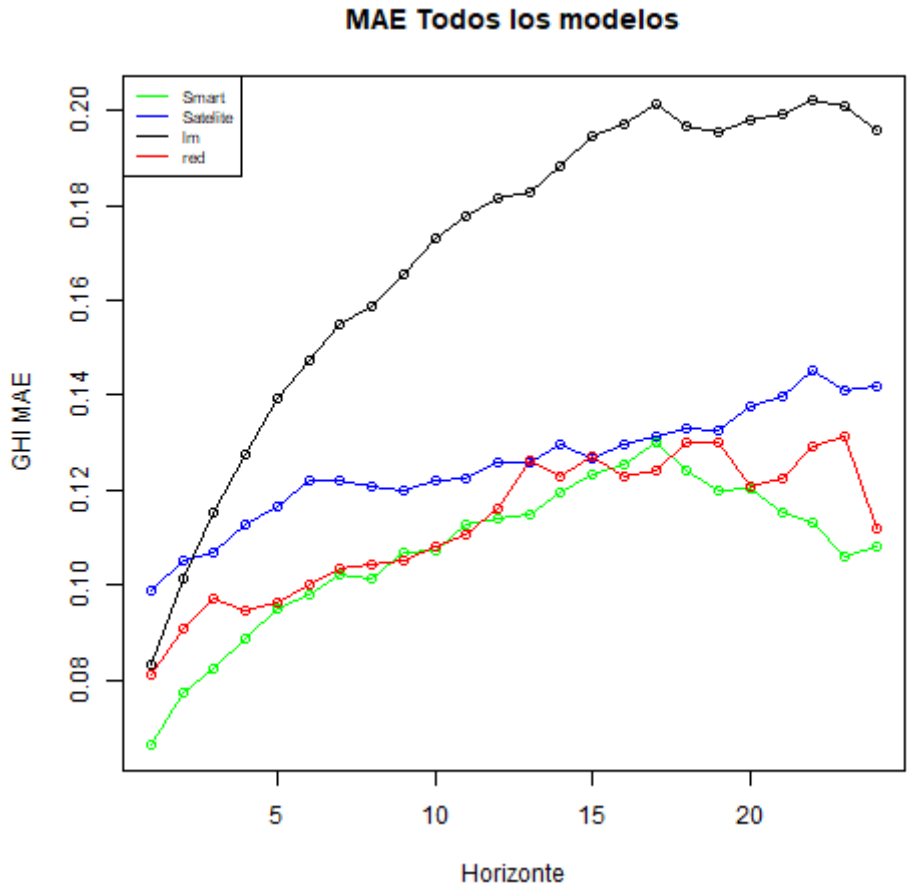

Fig. 4.6. MAE de todos los modelos de GHI para la cuarta semana

Tomando como referencia el error MAE de cada modelo en cada horizonte (ver imagen 4.6), se puede observar como para los experimentos realizados cogiendo la última semana de cada mes como conjunto de test, los modelos de redes neuronales construidos tienen un comportamiento parecido al obtenido por SmartPersistence entre los horizontes 13 y 19, obteniendo este último mejores resultados para horizontes cercanos y horizontes lejanos. Además se observa que la red poduce mejores resultados que Satélite.

En cuanto a los modelos lineales, se aprecia un buen comportamiento para instantes cercanos de tiempo, aunque el error cometido por estos crece cuanto más alejado esté el horizonte, presentando un peor comportamiento que la red de neuronas.

En el caso de tomar como referencia el RMSE cometido por los modelos en cada uno de los horizontes (ver figura 4.7), se aprecia como las redes neuronales poseen una mejor capacidad de generalización que el resto de modelos para todos los horizontes temporales. Si nos fijamos en el desempeño de los modelos lineales, se pueden sacar las mismas conclusiones que se obtuvieron para la medida de error MAE.

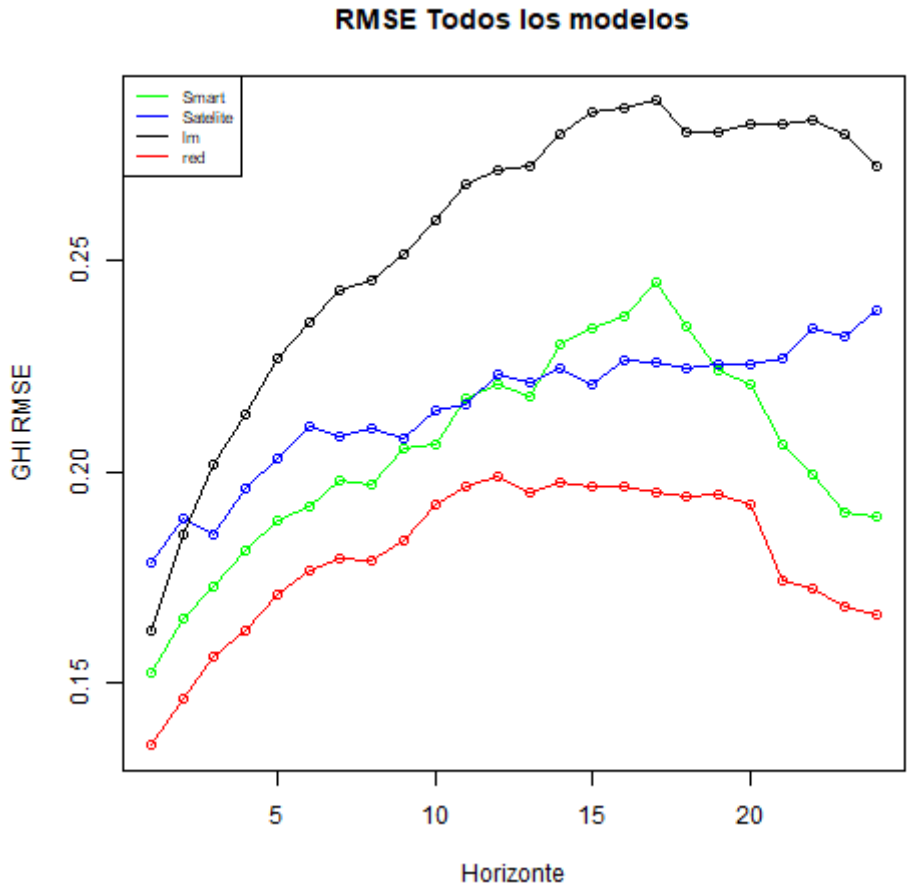

Fig. 4.7. RMSE de todos los modelos de GHI para la cuarta semana

Resultados para la tercera semana de Test

En la figuras 4.8 y 4.9 se muestra el MAE y el RMSE para todos los modelos cuando se utiliza la tercera semana de test, respectivamente.

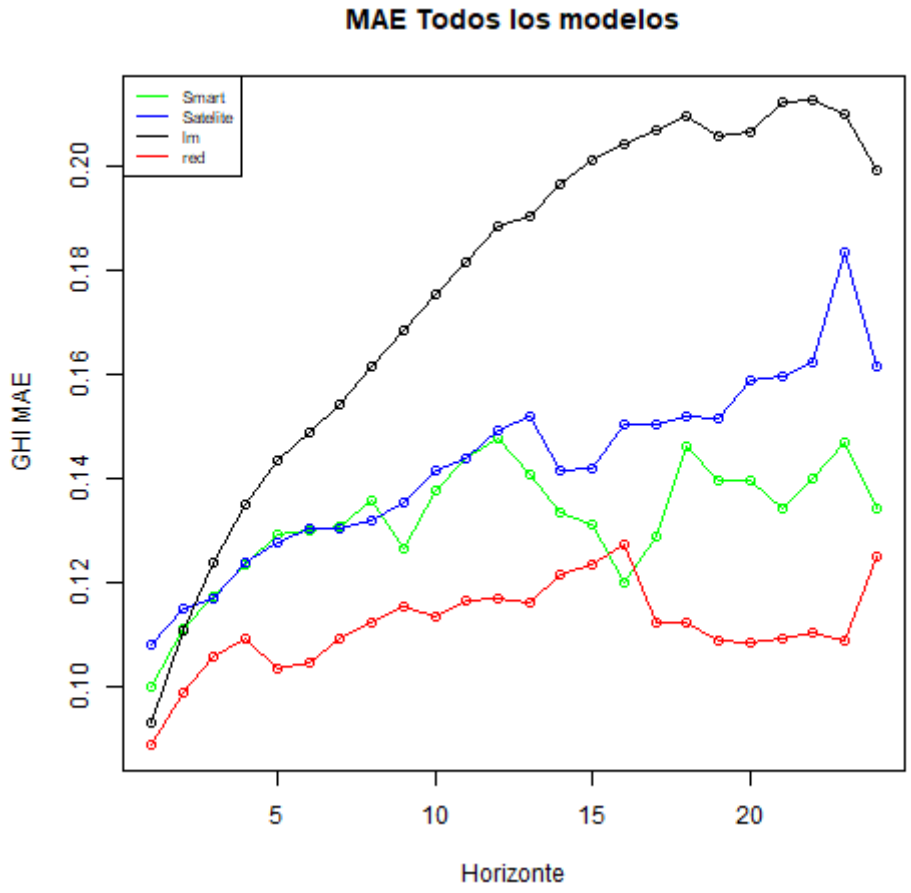

Fig. 4.8. MAE de todos los modelos de GHI para la tercera semana

Observando los errores MAE cometidos por los modelos para GHI expuestos en la gráfica anterior 4.8, se puede concluir que los modelos neuronales como norma general obtienen mejores resultados que el resto de modelos para cada uno de los horizontes. Las mismas conclusiones se pueden obtener si tomamos como referencia el error RMSE para la misma semana de test (ver figura 4.9).

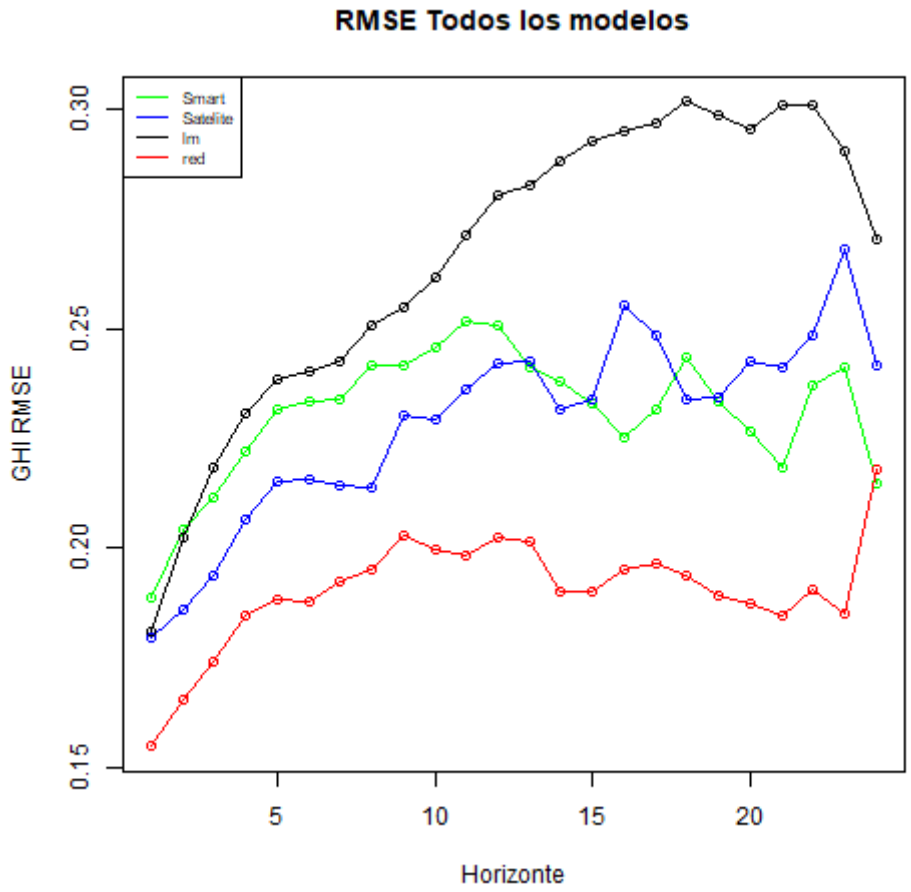

Fig. 4.9. RMSE de todos los modelos de GHI para la tercera semana

Por otro lado, tanto para MAE como para RMSE, los modelos lineales presentan un mal comportamiento, no solo comparado con la red, sino también con los modelos disponibles (SmartPersistence y Satélite), a excepción de los primeros horizonte (15 y 30 minutos) que es muy similar a SmartPesistence.

4.3.2. Resultados DNI

En las tablas 4.4 y 4.5 se presentan las configuraciones de hyper-parámetros de las redes que obtienen el menor error MAE en validación para DNI en cada horizonte temporal.

Horizonte	Topologia	RA	Ciclos
15	15	0,1	241
30	15	0,1	257
45	5	0,1	237
60	30	0,01	3294
75	5	0,05	431
90	50	0,1	180
105	30	0,005	6747
120	5	0,001	6732
135	25	0,005	4178
150	25	0,005	4637
165	25	0,005	5588
180	25	0,005	5906
195	20	0,01	2881
210	25	0,01	3044
225	10	0,01	2392
240	10	0,01	2614
255	10	0,01	2359
270	10	0,01	3216
285	40	0,2	4
300	50	0,2	6
315	50	0,2	8
330	50	0,2	12
345	5	0,005	6422
360	40	0,1	13

Tabla 4.4. Mejores modelos de DNI para la cuarta semana de Test

Horizonte	Topologia	RA	Ciclos
15	5	0,005	6980
30	5	0,005	6441
45	20	0,005	6311
60	5	0,005	6940
75	5	0,01	2335
90	5	0,005	4150
105	25	0,005	5305
120	5	0,005	3059
135	15	0,005	5073
150	15	0,005	5526
165	15	0,005	5543
180	10	0,01	2570
195	30	0,01	3340
210	30	0,01	3859
225	15	0,05	773
240	15	0,05	848
255	25	0,01	5473
270	10	0,1	388
285	50	0,2	2
300	50	0,2	2
315	50	0,2	2
330	10	0,01	6983
345	50	0,2	2
360	40	0,005	310

Tabla 4.5. Mejores modelos de DNI para la tercera semana de Test

Resultados para la cuarta semana de Test

En la figuras 4.10 y 4.11 se muestra el MAE y el RMSE para todos los modelos cuando se utiliza la cuarta semana de test, respectivamente.

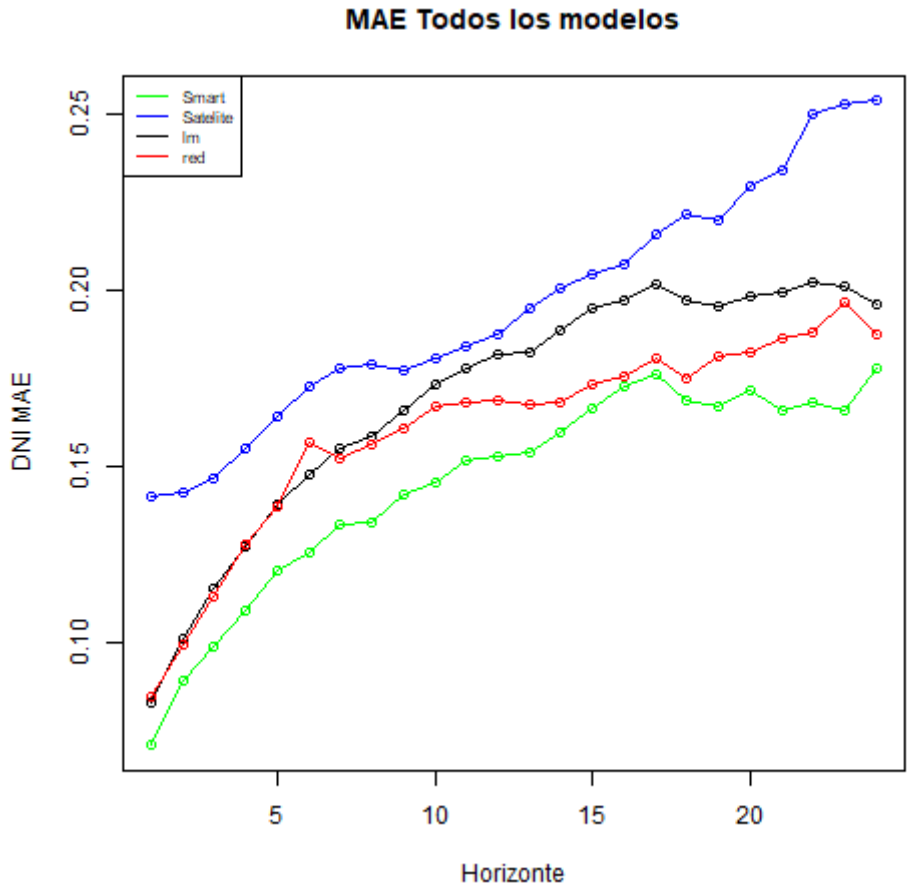

Fig. 4.10. MAE de todos los modelos de DNI para la cuarta semana

Teniendo en cuenta los errores MAE conseguidos por los modelos utilizando la cuarta semana para test (ver figura 4.10), se puede concluir que SmartPersistence desempeña un mejor papel a la hora de generalizar la variable DNI para todos los horizontes, mientras que las predicciones hechas con las redes neuronales obtienen una mayor tasa de error. Por otro lado, se aprecia como los modelos lineales siguen un comportamiento parecido al que habían desempeñado hasta ahora, es decir, producen buenos resultados a corto plazo, pero el error aumenta según aumenta el horizonte temporal.

Si se tiene en cuenta los resultados midiendo los errores RMSE (ver figura 4.11), se aprecia como las redes neuronales obtienen una buena generalización de la variable DNI, mientras que SmartPersistence produce peores resultados que los modelos lineales. En este caso, los modelos lineales presentan un comportamiento similar a las redes, aunque produciendo un peor error.

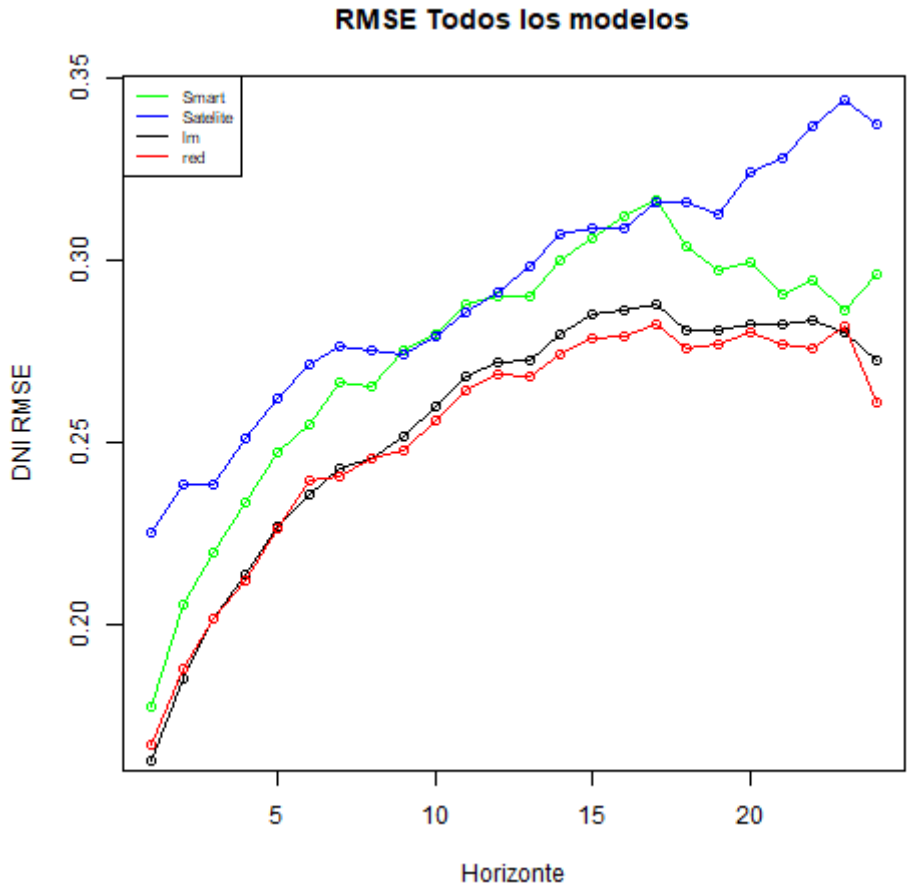

Fig. 4.11. RMSE de todos los modelos de DNI para la cuarta semana

Resultados para la tercera semana de Test

En la figuras 4.8 y 4.9 se muestra el MAE y el RMSE parara todos los modelos cuando se utiliza la tercera semana de test, respectivamente.

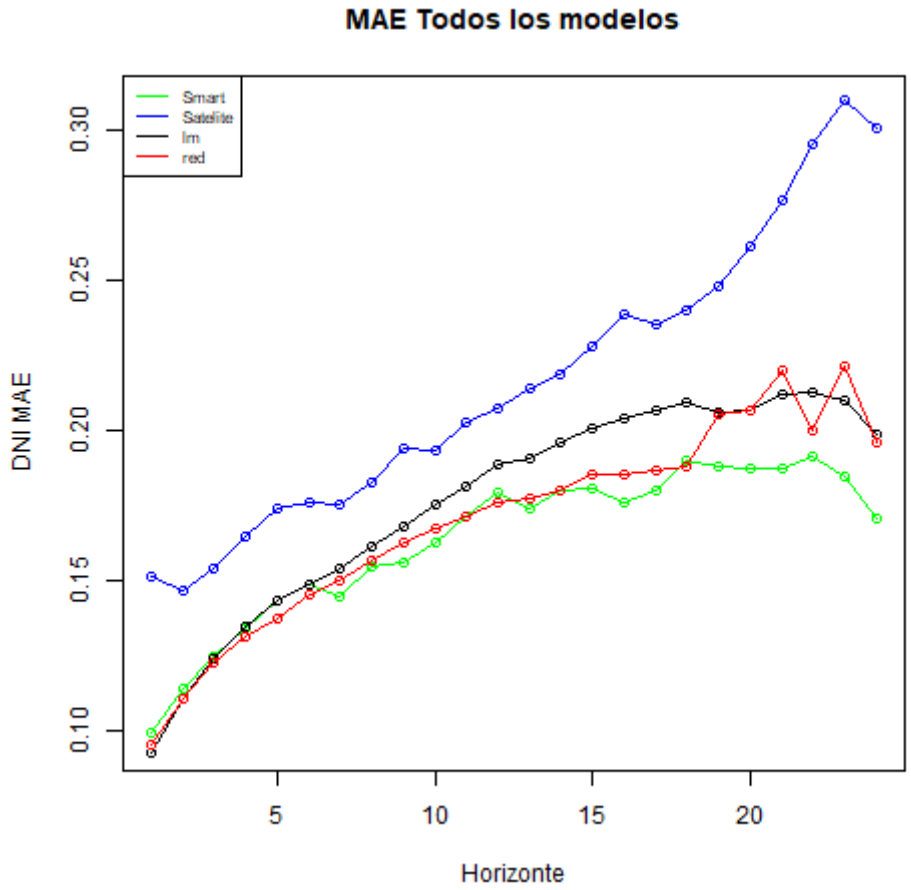

Fig. 4.12. MAE de todos los modelos de DNI para la tercera semana

En cuanto a los errores cometidos por los modelos a la hora de predecir los valores de DNI utilizando la tercera semana de cada mes de como conjunto de test, si utilizamos MAE como medida del error, el comportamiento es similar a lo visto en el apartado anterior: SmartPersistence desempeña un mejor papel a la hora de generalizar la variable DNI para los horizontes lejanos (horizontes 19 a 24), mientras que las predicciones hechas con las redes neuronales obtienen una mayor tasa de error. Para el resto de los horizontes, SmartPersistence y los modelos neuronales están bastante igualados. Los modelos lineales son peores que los modelos neuronales en los horizontes 4 y 5, presentando para el resto un comportamiento similar.

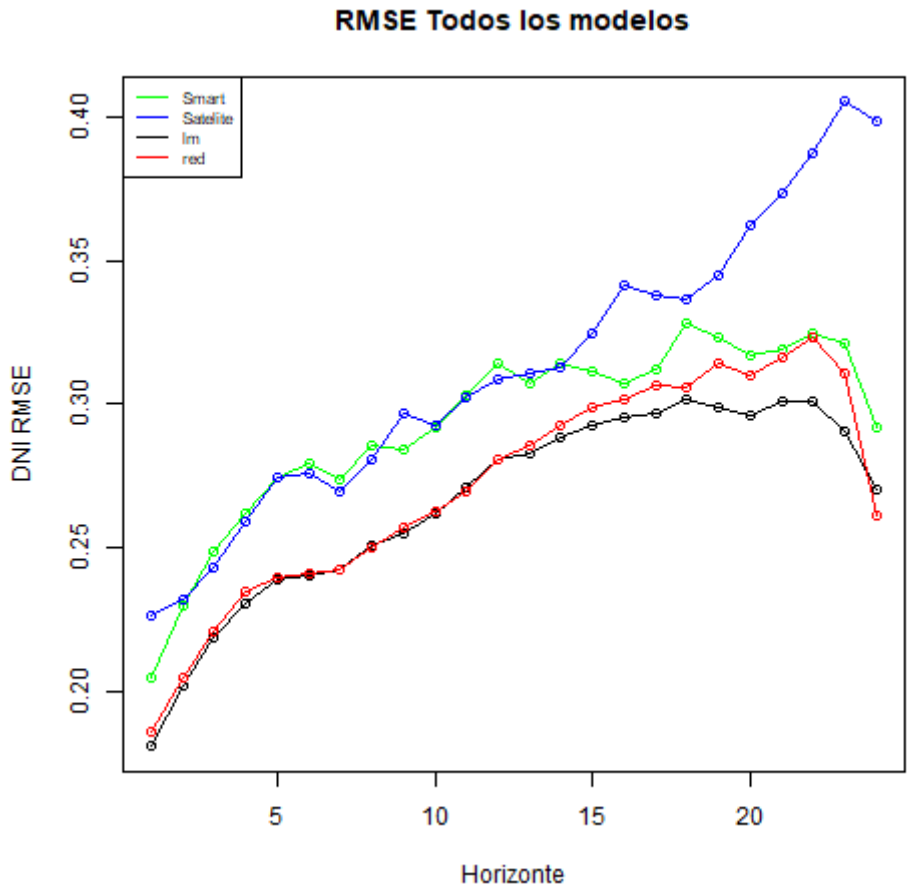

Fig. 4.13. RMSE de todos los modelos de DNI para la tercera semana

Si se toma RMSE como medida del error para este bloque de test, las redes neuronales obtienen una buena generalización de la variable DNI, mientras que SmartPersistence produce peores resultados que los modelos lineales. En este caso, se puede observar como los modelos lineales y los Perceptrones Multicapa se comportan de manera muy similar, llegando los modelos lineales a producir mejores resultados desde los horizontes 13 al 23.

4.4. Resumen de los resultados

Como se ha observado en los resultados anteriormente presentados, en general las redes de neuronas proporcionan buenos resultados superando en muchos casos a los modelos disponibles y a los modelos lineales, pero el comportamiento puede variar dependiendo de la semana de test y del tipo de error que se esté utilizando.

Dicha diferencia en el comportamiento de MAE y RMSE es debido a que MAE mide la diferencia en valor absoluto, mientras que RMSE mide la diferencia al cuadrado. Por lo que, si para unos pocos patrones, la diferencia es grande, al elevarla al cuadrado se

hace más grande y eso puede hacer que esos patrones tengan mucho peso en el cálculo de RMSE frente a los demás, dando lugar a un error alto en RMSE y no tanto en MAE.

Por tanto, para tener una visión global, se evalúa la media de los errores para todos los horizontes. En las tablas 4.6 y 4.7 se muestran los errores medios en GHI para todos los horizontes para cada una de las semanas de test utilizadas.

Cuarta Semana	**SmartPersistence**	**Satélite**	**Perceptron Multicapa**	**Lm**
MAE medio	0,107134922	0,125398302	0,112739241	0,16989521
RMSE medio	0,205197465	0,215358389	0,179936919	0,255798347

Tabla 4.6. Media de los errores de GHI para la cuarta semana de test

Tercera Semana	**SmartPersistence**	**Satélite**	**Perceptron Multicapa**	**Lm**
MAE medio	0,131914145	0,142424943	0,111537052	0,176609809
RMSE medio	0,23089198	0,228501817	0,190190136	0,266170028

Tabla 4.7. Media de los errores de GHI para la tercera semana de test

Como se puede observar, para GHI, los modelos de redes neuronales, en media, son los que mejores resultados producen, excepto para la cuarta semana de test, teniendo en cuanta el error MAE. En este caso, SmartPersistence obtiene un error medio menor, aunque el de las redes sigue siendo bastante cercano.

Por otro lado, se observa como los modelos neuronales obtienen mejores resultados medios que Satélite y los modelos lineales para todos los casos.

En cuanto a DNI, En las tablas 4.8 y 4.9 se muestran los errores medios obtenidos para todos los horizontes obtenidos por cada uno de los modelos.

Cuarta Semana	**SmartPersistence**	**Satélite**	**Perceptron Multicapa**	**Lm**
MAE medio	0,145252912	0,125398302	0,160632552	0,16989521
RMSE medio	0,2748309	0,291924604	0,252811163	0,255798347

Tabla 4.8. Media de los errores de DNI para la cuarta semana de test

Tercera Semana	SmartPersistence	Satélite	Perceptron Multicapa	Lm
MAE medio	0,163530406	0,216217814	0,170047578	0,176609809
RMSE medio	0,292773402	0,312286831	0,271449092	0,266170028

Tabla 4.9. Media de los errores de DNI para la tercera semana de test

Tomando MAE como medida de error de los modelos de DNI, se puede apreciar como SmartPersistence presenta los mejores resultados en media para ambas semanas de test, seguido de las redes neuronales. Por el contrario, observando RMSE, los menores errores medios son proporcionados por las redes neuronales.

Por otro lado, se observa como para RMSE los modelos lineales no proporcionan los peores resultados, llegando a estar igualados a las redes de neuronas en ciertas ocasiones. Para todos los casos, Satélite aporta los peores resultados.

Capítulo 5

Findings and future related studies

Finally, in this chapter the final conclusions as well as the different difficulties faced along the development of the solar irradiation forecasting will be presented. Some future strands of work will be mentioned too.

5.1. Findings

The main conclusions drawn about the achived results are:

- Comparing the results obtained with the neural models with the models treated in the project, it can be seen that the networks obtained a similar performance to the SmartPersistece model, even improving it in some cases. On the other hand, Multilayer Perceptrons improve, in most cases, the results obtained by Satellite.
- In general, neural networks result in better prediction approaches for global solar radiation values. For direct irraciation forecasting problems, neural networks get worse behaviours.
- A great dependence was found between the results obtained and the dates selected to split the data into training, validation and test sets, showing so the sensitiveness of the results when choosing the test data. Whithin the related problem this sensitivity is due to the fact that the climate in the same week of the same month may be different for different years. Therefore, it has been necessary to choose different test weeks.
- Making reference to the previous point and looking at the tables of the average errors of each model shown in Chapter 4 (see tables 4.6, 4.7 4.8, 4.9) it has been observed that, as a general rule, lower errors (on average) are obtained when the fourth week of each month is taken as a test set.

In addition to the previous ones, other opinions were extracted along the Thesis development:

- It has been proven that neural networks are a very powerful tool in order to build models from data that may belong to different domains. However, when generating forecasting models, a great knowledge of the domain is required if these models are to obtain accurate results.
- It has been possible to know the R language thoroughly, which is a tool that, in a first contact may seem unfriendly, but once you have taken a certain practice it is a very powerful tool which, in the machine learning learning and data analysis domain, you can perform complex tasks in short time and in a simple way with.
- The prediction problem addressed in the thesis belongs to a very complex domain where it is very difficult to reach accurate results. This is due to the fact that so many factors have direct influence on the weather.

5.2. Future works

The results obtained using radiation models based on neural networks, although in some cases improve Smart Persistence and Satellite models behaviour, are not as good as expected. For this reason, some improvements or possible future works are presented:

- It would be interesting to see how models using radial basis functions behave. Specifically, support vector machines with radial function could be generated to compare the results.
- Due to the bias introduced by the choice of a week in a specific month as a test set, another type of validation techniques could be performed to see the networks behavior. For example, cross validation using four weeks of each month (four folds or groups) and calculate the means. Even cross validation could be used using 5 days of each month, resulting in 6 folds.
- In addition to the solar radiation measurements in Seville, data from meteorological stations located in three different points of the peninsula are available: Madrid, Jaén and Lisbon. With this, it would be interesting to study the generalization capacity of the neural models using these data sets.

Bibliografía

[1] H. M. Diagne, P. Lauret y M. David, "Solar irradiation forecasting: state-of-the-art and proposition for future developments for small-scale insular grids", en *WREF 2012-World Renewable Energy Forum*, 2012.

[2] V. Lara-Fanego, J. Ruiz-Arias, D. Pozo-Vázquez, F. Santos-Alamillos y J. Tovar-Pescador, "Evaluation of the WRF model solar irradiance forecasts in Andalusia (southern Spain)", *Solar Energy*, vol. 86, n.º 8, pp. 2200-2217, 2012.

[3] C. Voyant et al., "Machine learning methods for solar radiation forecasting: A review", *Renewable Energy*, vol. 105, pp. 569-582, 2017.

[4] G. Terren-Serrano, "Machine Learning Approach to Forecast Global Solar Radiation Time Series", 2016.

[5] A. Mellit y A. M. Pavan, "A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid-connected PV plant at Trieste, Italy", *Solar Energy*, vol. 84, n.º 5, pp. 807-821, 2010.

[6] N. Sharma, P. Sharma, D. Irwin y P. Shenoy, "Predicting solar generation from weather forecasts using machine learning", en *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, IEEE, 2011, pp. 528-533.

[7] J.-L. Chen, H.-B. Liu, W. Wu y D.-T. Xie, "Estimation of monthly solar radiation from measured temperatures using support vector machines-a case study", *Renewable Energy*, vol. 36, n.º 1, pp. 413-420, 2011.

[8] T. Chen y C. Guestrin, "Xgboost: A scalable tree boosting system", en *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, ACM, 2016, pp. 785-794.

[9] B. Wolff, E. Lorenz y O. Kramer, "Statistical learning for short-term photovoltaic power predictions", en *Computational sustainability*, Springer, 2016, pp. 31-45.

[10] Y. Gala, A. Fernández y J. Dorronsoro, "Machine learning prediction of global photovoltaic energy in spain", en *International Conference on Renewable Energies and Power Quality*, 2014, p. 278.

[11] R. Aler, R. Martín, J. M. Valls e I. M. Galván, "A study of machine learning techniques for daily solar energy forecasting using numerical weather models", en *Intelligent Distributed Computing VIII*, Springer, 2015, pp. 269-278.

[12] J. H. Friedman, "Greedy function approximation: a gradient boosting machine", *Annals of statistics*, pp. 1189-1232, 2001.

[13] J. Huertas-Tato et al., "Predicting Global Irradiance Combining Forecasting Models Through Machine Learning", en *International Conference on Hybrid Artificial Intelligence Systems*, Springer, 2018, pp. 622-633.

[14] E. Turban, R. Sharda y D. Delen, *Decision Support and Business Intelligence Systems*. Prentice Hall Learning Outcomes, 2010.

[15] J. Villavicencio, "Introducción a series de tiempo", *Obtenido de Sitio web del Instituto de Estadísticas de Puerto Rico*, 2010.

[16] M. G. Grolemund y H. Wickham, "Package lubridate", 2013.

[17] M. Dowle et al., "Package data. table", 2018.

[18] H. Wickham y R. Francois, "dplyr: A grammar of data manipulation", *R package version 0.4*, vol. 3, 2015.

[19] C. N. Bergmeir y J. M. Benítez Sánchez, "Neural networks in R using the Stuttgart neural network simulator: RSNNS", American Statistical Association, 2012.

[20] H. Wickham, *ggplot2: elegant graphics for data analysis*. Springer, 2016.

[21] C. Cortes y V. Vapnik, "Support-vector networks", *Machine learning*, vol. 20, n.º 3, pp. 273-297, 1995.

[22] R. Martin, R. Aler, J. M. Valls e I. M. Galván, "Machine learning techniques for daily solar energy prediction and interpolation using numerical weather models", *Concurrency and Computation: Practice and Experience*, vol. 28, n.º 4, pp. 1261-1274, 2016.

[23] R. H. Inman, H. T. Pedro y C. F. Coimbra, "Solar forecasting methods for renewable energy integration", *Progress in energy and combustion science*, vol. 39, n.º 6, pp. 535-576, 2013.

[24] A. Sharma y A. Kakkar, "Forecasting daily global solar irradiance generation using machine learning", *Renewable and Sustainable Energy Reviews*, 2017.

[25] R. Perez et al., "Validation of short and medium term operational solar radiation forecasts in the US", *Solar Energy*, vol. 84, n.º 12, pp. 2161-2172, 2010.

[26] P. Isasi Vinuela e I. Galván León, "Redes de neuronas artificiales", *Un Enfoque Práctico, Editorial Pearson Educación SA Madrid España*, 2004.

[27] B. Lantz, *Machine learning with R*. Packt Publishing Ltd, 2015.